Proyecto Fin de Carrera
Ingeniería Informática

NAVEGACIÓN VISUAL USANDO UNA DESCRIPCIÓN DE LA RUTA CON SECUENCIAS DE IMÁGENES

Eduardo Montijano Muñoz

Director: Carlos Sagüés Blázquiz

Departamento de Informática e Ingeniería de Sistemas
Centro Politécnico Superior
Universidad de Zaragoza

Septiembre 2007

Índice general

Índice de figuras

Índice de tablas

Capítulo 1
Objeto y alcance

1.1. Introducción

Desde el principio de su existencia, el hombre siempre se ha propuesto inventar artefactos que simplifiquen las tareas que realiza en su día a día. Los avances de la ciencia en los últimos años han permitido el desarrollo de máquinas que puedan realizar tareas complejas de manera autónoma, y entre todas ellas se debe destacar a los robots.

Las tareas que realizan en la actualidad los robots suelen ser sencillas y de carácter repetitivo. En muchos casos, además, se trata de tareas que se realizan en espacios interiores, lo que implica que los entornos en los que se va a mover son poco cambiantes. Algunas de ellas requieren que un robot móvil repita constantemente un camino que ha aprendido para llevar y traer objetos. Un ejemplo de estas características podría ser un robot cartero, en el seno de una empresa. El robot repite todos los días la misma ruta para entregar las cartas a sus destinatarios. Los robots reales distan mucho de parecerse a los descritos en novelas o películas de ciencia ficción, máquinas pensantes con alta capacidad de raciocinio. Actualmente, en el marco de investigación y realizaciones de prototipos experimentales existen robots con capacidad de realizar algunas tareas sencillas en las que tanto el movimiento como la percepción se llevan a cabo de forma autónoma.

Los robots deben entenderse como herramientas que simplifiquen la vida de los humanos. Tareas que podrían suponer un riesgo potencial para las personas pueden ser ejecutadas por robots de manera automática, minimizando así los riesgos. Entre estas tareas se podrían destacar el transporte de objetos pesados o sustancias tóxicas, o tareas relacionadas con la exploración de un terreno desconocido. Naturalmente, a los robots también se les pueden asignar tareas domésticas que, aunque no impliquen un riesgo para nuestra seguridad, consigan hacer nuestras vidas más agradables.

Existen múltiples formas de realizar trayectorias que debe seguir un robot; las más sencillas son las que utilizan una planificación estática con la hipótesis de que el entorno no cambia y es conocido. Otras, más complejas, emplean planificación dinámica y calculan los objetivos dependiendo de la información que obtienen en tiempo real del medio en que

se encuentran.

Suele ocurrir que, independientemente del tipo de planificación que se use, el robot no consigue llegar a su destino. La principal causa es la falta de precisión de los sensores de la odometría (posición y orientación respecto a las coordenadas iniciales) del robot. Esto puede ser debido a que no tienen en cuenta la fuerza de rozamiento, la utilización de velocidades muy grandes o muy pequeñas o a otras causas ajenas al robot tales como pequeños obstáculos encontrados en el camino, etc...

Para corregir estos errores debe recurrirse a técnicas adicionales de percepción del entorno que aporten información más precisa de la localización exacta del robot. La generación de mapas con sensores láser o el uso de técnicas de visión por computador son algunos ejemplos de métodos que se utilizan en la actualidad para reducir los errores cometidos por los sensores odométricos.

Para que el diseño de estas técnicas de percepción sea el adecuado hay que considerar varios aspectos. Así, si se pretende que las correcciones puedan realizarse en tiempo real habrá que conseguir que el método utilizado sea eficiente; por otro lado, para que el error sea lo más pequeño posible necesitaremos garantizar que nuestro método sea robusto. Muchas veces, que un sistema sea robusto implica que no sea eficiente, ya que necesita una gran cantidad de cálculos adicionales para garantizar la calidad de los resultados que obtiene. Conseguir un equilibrio entre eficiencia y robustez puede resultar complicado y es uno de los principales objetivos de los investigadores a la hora de diseñar nuevos métodos para la realización autónoma de tareas de movimiento o percepción.

En este proyecto se estudian algunos métodos que utilizan la visión por computador como medio para corregir errores en la repetición de trayectorias, analizando las ventajas e inconvenientes de las distintas técnicas utilizadas y proponiendo mejoras en las mismas que las hagan más eficientes y robustas.

1.2. Objetivos

El objetivo principal de este proyecto ha sido la mejora en los algoritmos de control de movimientos del robot, dando lugar a comportamientos más fiables en la repetición de trayectorias aprendidas previamente utilizando técnicas de visión por computador.

Cumplir este objetivo ha implicado la realización de un estudio comparativo de varias técnicas de extracción de características de imágenes en el ámbito de la navegación de un robot en un entorno cerrado que utiliza imágenes de referencia. Para realizar este estudio se ha diseñado un amplio conjunto de experimentos que permitan comparar los métodos seleccionados a diferentes niveles. Estos experimentos evaluarán la eficacia de los métodos, su coste computacional y la robustez que presentan en distintas situaciones.

La mayoría de las pruebas se han realizado sobre uno de los robots *Pioneer* que posee el Grupo de Robótica, Percepción y Tiempo Real de la Universidad de Zaragoza. En el apéndice A se explican en detalle las características de este robot.

Los extractores elegidos para realizar la comparación son el extractor de contornos verticales diseñado por Burns [7], un método propuesto hace varios años que siempre ha tenido gran aceptación, y un método de extracción de puntos publicado recientemente llamado SURF (Speeded Up Robust Features) [6] que está teniendo una gran repercusión en el ámbito de la visión por computador por la calidad de las características que extrae de las imágenes.

Las correcciones en los movimientos del robot se pueden realizar considerando imágenes tomadas en dos instantes distintos. Analizando las características que se extraen de cada una de las dos imágenes se puede obtener información acerca del movimiento que debe realizar el robot para ir desde la posición correspondiente a la primera imagen hasta la posición correspondiente a la segunda. Los detalles técnicos de este proceso se explican en la memoria.

Adicionalmente se ha estudiado la influencia que tiene la técnica utilizada para realizar los emparejamientos y su robustez en distintas situaciones.

1.3. Alcance

La información de partida para el desarrollo del proyecto ha consistido en :

- Un programa para la extracción de contornos verticales de una imagen desarrollado por el Grupo de Robótica, Percepción y Tiempo Real.
- Un programa para la obtención de transformaciones proyectivas entre dos imágenes a partir de los emparejamientos de rectas verticales en ambas imágenes, también desarrollado por el Grupo de Robótica, Percepción y Tiempo Real.
- Códigos de Matlab para dibujar un robot esquemático visto desde arriba en la posición deseada.
- Librerías Player/Stage, versión 2.03, de código abierto, que facilitan la toma de imágenes o la realización de desplazamientos con el robot.
- Archivos objeto, precompilados, que contienen las funciones necesarias para la extracción de las características SURF de una imagen desarrollados por Herbert Bay y colaboradores [6], del *Computer Vision Laboratory* del ETH de Zurich.

A lo largo del desarrollo del proyecto se han realizado las siguientes acciones:

Dado que el software mencionado anteriormente no estaba diseñado específicamente para los robots que se encuentran en el laboratorio, ha sido necesaria una labor previa de adaptación de los códigos disponibles para poder usarlos correctamente en el robot *Pioneer*, realizando diversos experimentos para comprobar su correcto funcionamiento.

Se han modularizado los códigos desarrollados por el Grupo de Robótica, Percepción y Tiempo Real. De esta forma, se han podido incorporar los nuevos métodos de emparejamiento al sistema y la posibilidad de extraer contornos verticales o puntos SURF. Se ha organizado todo para que en el futuro sea posible la adición de nuevos módulos de visión de manera sencilla y también se puedan añadir sin dificultad otros métodos de emparejamiento robusto. En definitiva se ha buscado que el código sea reusable en su totalidad y fácilmente escalable.

Con todo el código ya incorporado se ha procedido a realizar experimentos de diversa índole para verificar tanto el correcto funcionamiento del sistema, como la eficiencia y la calidad de los resultados obtenidos por el mismo.

Los primeros experimentos realizados han sido de carácter estático. Se ha observado como se comportan los diferentes métodos de emparejamiento propuestos, primero con datos de simulación generados aleatoriamente y después con datos extraídos de imágenes tomadas por el robot.

Se ha diseñado e implementado un programa de aprendizaje para el robot. Este programa permite al usuario mover libremente al robot por el entorno. El robot va capturando imágenes y almacenando la información necesaria que le permita repetir posteriormente el camino indicado sin ayuda externa.

Se ha desarrollado un programa de seguimiento de trayectorias que emplea las características obtenidas durante el aprendizaje para que el robot sea capaz de repetirlas sin ayuda externa. Se han realizado experimentos sencillos para comprobar que el robot es capaz de efectuar correcciones en la traslación y rotación; estas correcciones hacen que la trayectoria que sigue el robot se acerque más a la trayectoria deseada Se han probado diversas soluciones y analizado las ventajas e inconvenientes de cada una de ellas.

Los algoritmos empleados en estos programas se han inspirado tanto en algoritmos basados en la literatura como en algoritmos diseñados durante el desarrollo del proyecto.

En la etapa final se han realizado algunos experimentos de mayor complejidad observando las diferencias entre los distintos métodos testados.

Al final todo el sistema ha quedado integrado en un único programa de manejo sencillo que permita a futuros investigadores su uso y ampliación.

1.4. Estructura de la memoria

El capítulo 2 de la memoria se ha dedicado a describir el funcionamiento de los distintos métodos de extracción y emparejamiento de características de las imágenes y las técnicas requeridas para el cálculo de matrices de homografía.

En el capítulo 3 se explica todo el proceso que sigue el robot para realizar la navegación utilizando la visión por computador.

Los capítulos cuatro y cinco contienen la información acerca de los experimentos realizados y los resultados que se han obtenido en ellos.

El sexto capítulo expone las conclusiones que se han obtenido al finalizar los experimentos, las dificultades encontradas y las posibles ampliaciones que se pueden hacer en el futuro partiendo del trabajo realizado en el proyecto.

Por último, en el capítulo siete se comentan los principales hitos temporales en la realización del proyecto y se muestra el diagrama de Gant, que resume como se ha distribuido el tiempo.

Tras la memoria se incluyen varios apéndices que recogen información más detallada de algunos aspectos tratados en el proyecto como la extracción de características o las técnicas de emparejamiento robusto estudiadas.

Capítulo 2
Técnicas de visión por computador

La vista es uno de los sentidos más importantes para el ser humano. El cerebro humano es capaz de detectar colores, contornos y zonas de interés de una simple imagen y relacionarlos rápidamente con la realidad.

Para un computador estas acciones, que resultan tan cotidianas en la vida de las personas, suponen todo un reto. En una imagen de computador, lo único que se observa es un patrón bidimensional de brillo. Para poder extraer información del mismo, es preciso realizar el procesamiento de la imagen centrando el interés en regiones o contornos de la misma. Las técnicas de extracción de características tienen por objeto encontrar, en una imagen, un conjunto de datos que la caractericen de la mejor forma posible y permitan establecer relaciones entre imágenes. Estas técnicas de extracción de características de una imagen suelen ser complejas y de alto coste computacional.

Para el desarrollo de este proyecto se han estudiado diversas técnicas extracción de características. Antes de realizar ningún experimento, se han descartado algunas de ellas, ya que se ha detectado que no iban a producir resultados deseados. Las técnicas que se han descartado se explican en el apéndice B, donde también se comentan los detalles por los que han sido descartadas.

A continuación se presentan dos técnicas de extracción de características de imágenes tomadas con una cámara digital. Posteriormente se explica qué tratamiento se realiza con dicha información para extraer información de movimiento relativo entre las imágenes obtenidas del entorno.

2.1. Características de las imágenes

2.1.1. Contornos verticales

En un entorno cerrado y creado por el hombre, es fácil observar que existen una gran cantidad de líneas verticales. Si nuestro sistema de visión es capaz de identificarlos, pueden

resultar una gran fuente de información para ayudar a los robots a realizar trayectorias de forma precisa. Un método que se utiliza para obtener los contornos verticales de una imagen es el método de Burns [7].

Este método se basa en la agrupación de los píxeles según la orientación de su gradiente. En primer lugar, se calcula para cada píxel el módulo y orientación del gradiente. Para esto, en lugar de utilizar las derivadas, se emplea una máscara de convolución. Una vez que se tiene la orientación del gradiente en cada pixel, se procede a la agrupación de éstos en "regiones de soporte de recta" (LSR) mediante un análisis de conectividad.

En la siguiente etapa, el extractor realiza un filtro, eliminando todos aquellos blobs (conjuntos conexos de píxeles pertenecientes a la misma LSR) obtenidos en el análisis de conectividad que no tengan la orientación adecuada o cuyo número de píxeles sea inferior a un umbral determinado. Con los blobs que cumplen todas las restricciones impuestas se obtienen rectas mediante aproximación por mínimos cuadrados. Un último filtro elimina aquellas rectas que no alcancen una longitud mínima.

Una vez que se han obtenido las rectas de dos imágenes, es posible realizar una comparación de ellas buscando emparejamientos entre los conjuntos de rectas de ambas imágenes. Un emparejamiento se puede definir como una correspondencia entre una característica de la primera imagen y otra de la segunda que tienen valor de los descriptores que las definen similares. Las técnicas básicas de emparejamiento se detallan en el apéndice C.

Para que los emparejamientos realizados entre dos conjuntos de rectas sean más robustos, además de almacenar las coordenadas inicial y final de la recta, se guarda también su orientación y datos del contraste y del nivel medio de gris.

La figura 2.1 muestra un ejemplo de los contornos obtenidos utilizando el extractor de contornos verticales.

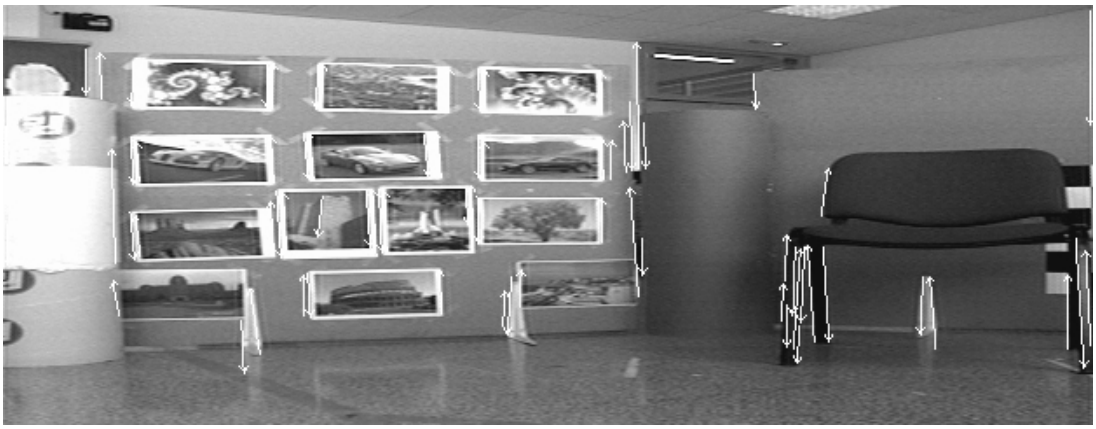

Figura 2.1: Ejemplo de una imagen capturada por el robot con los contornos verticales extraídos mediante el método de Burns.

2.1.2. Características SURF

Otra característica de una imagen que también se ha usado en el contexto de la visión 2D para obtener estructura y movimiento es conocida como SURF (Speeded Up Robust Features). En este caso en lugar de extraer rectas, como se hacía en el método de Burns, lo que se extraen son regiones que cumplen unas determinadas propiedades. A continuación se explica de forma resumida el método. Se ha adjuntado una explicación más detallada acerca de estas características novedosas en el apéndice B.

Para encontrar puntos de interés en una imagen cualquiera el método propuesto por Herbert Bay [6] busca máximos locales en el determinante de la matriz Hessiana

$$\mathcal{H}(p,\sigma) = \begin{pmatrix} L_{xx} & L_{xy} \\ L_{xy} & L_{yy} \end{pmatrix} \tag{2.1}$$

donde L_{xx}, L_{xy} y L_{yy} representan las derivadas segundas parciales de la gaussiana $G(\sigma)$ respecto a las coordenadas de cada pixel $p = (x, y)$. Para realizar los cálculos de manera más eficiente el extractor emplea imágenes integrales y filtros de caja en lugar de calcular las derivadas, como se explica en el apéndice B.

Una vez que se han detectado los puntos en los que el valor del determinante es un máximo local se procede a calcular un descriptor que sirva para identificarlos de manera unívoca respecto al resto de puntos. La longitud de este descriptor es configurable por el usuario y ha sido uno de los puntos que se han estudiado en el apartado de experimentos.

En la figura 2.2 se observan las características obtenidas con este extractor para la misma imagen que la presentada en la figura 2.1.

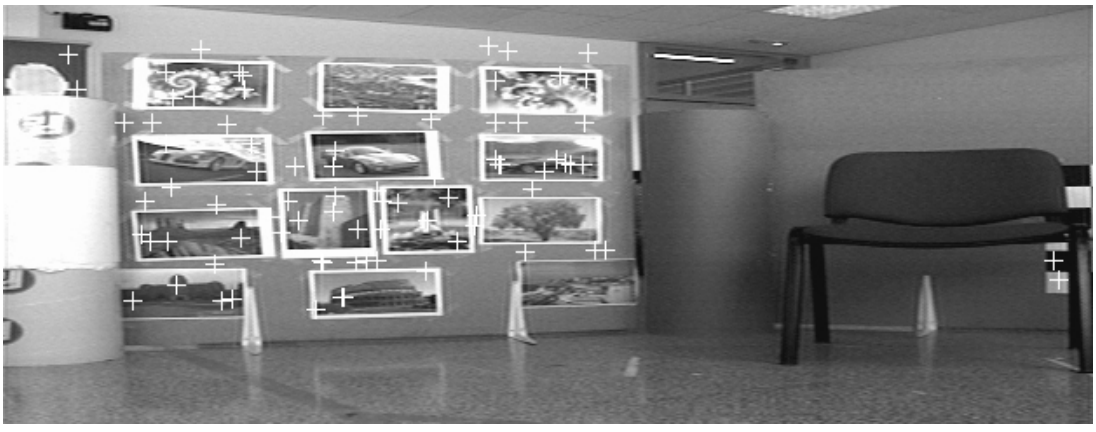

Figura 2.2: Ejemplo de una imagen capturada por el robot con las características SURF obtenidas.

2.2. Emparejamientos: emparejamiento robusto

Una vez que se han extraído las características de las imágenes se emplea un emparejador para encontrar correspondencias entre ambas. La distancia de Mahalanobis o la norma euclídea son dos funciones que se utilizan habitualmente para calcular los emparejamientos. Los métodos empleados en este proyecto para el emparejamiento básico o inicial son muy utilizados en el campo de la visión por computador y se detallan en el apéndice C

Con los emparejamientos obtenidos se plantea un sistema de ecuaciones que permite obtener una restricción geométrica entre las imágenes que se usa para obtener mejores emparejamientos y, posteriormente, para calcular el movimiento relativo entre las imágenes.

Suele ocurrir que, dentro del conjunto de emparejamientos, hay un porcentaje de ellos que son espurios (véase figura 2.3) . Si se resuelve el sistema sobredimensionado sin eliminarlos, la solución que se obtiene no es la correcta. Para que la matriz de homografía calculada sea correcta hay que conseguir eliminar los emparejamientos espurios del conjunto de emparejamientos obtenido, este proceso se conoce como emparejamiento robusto. Existen diversas técnicas de emparejamiento robusto que consiguen eliminar los emparejamientos no deseados. En este proyecto se han realizado experimentos con dos métodos de emparejamiento robusto probabilísticos.

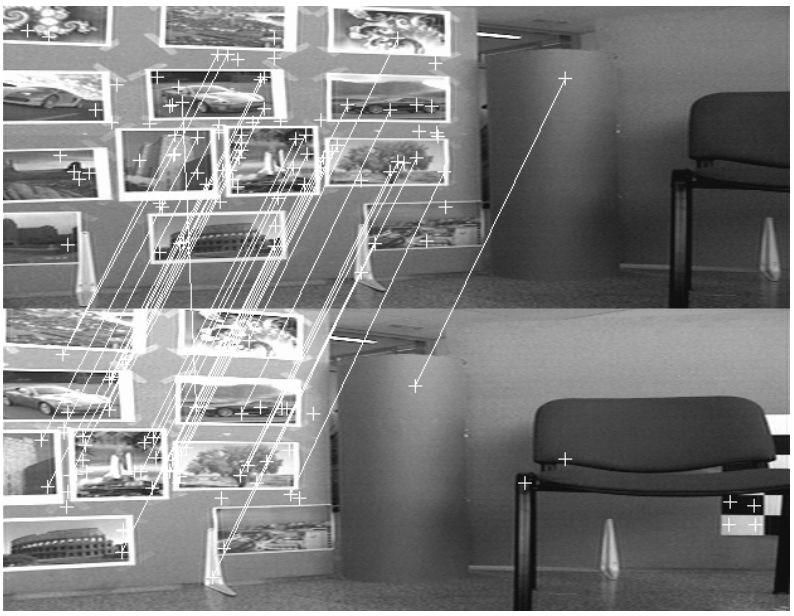

Figura 2.3: Ejemplo de emparejamientos entre puntos SURF extraídos de dos imágenes. En la imagen se pueden apreciar algunos emparejamientos espurios.

2.2.1. Método RANSAC

El método RANSAC (RANdom SAmple Consensus) [14] es un método de emparejamiento robusto que da como resultado la solución más votada de entre unas cuantas, calculadas a partir de conjuntos mínimos obtenidos aleatoriamente. Para calcular una homografía en dos dimensiones se necesitan un mínimo de tres emparejamientos. La idea del método RANSAC consiste en seleccionar un número de subconjuntos de tres emparejamientos elegidos aleatoriamente dentro del total, que garantice que, con una probabilidad superior al 99 %, al menos uno de dichos subconjuntos contendrá sus tres emparejamientos correctos, es decir, que ninguno de ellos es un espurio.

Para cada uno de los subconjuntos formados se calcula la homografía. Esta homografía se valora mediante un sistema de votaciones. Para ello, se fija un umbral que separe los emparejamientos buenos de los malos para esta homografía. Un emparejamiento se considerará bueno si y sólo si la homografía lleva una característica a la otra con un error menor que el umbral considerado. Se contabiliza el número de emparejamientos buenos (votos favorables), quedándonos al final con la homografía que más votos haya tenido.

Una vez que se ha elegido la homografía más votada como solución, se consideran emparejamientos buenos aquellos que hayan votado a dicha solución, mientras que los espurios serán los que no la hayan votado.

Para reducir el coste computacional, en el momento en que una homografía recibe más del 95 % de los votos, esta se considera solución definitiva. De esta forma en conjuntos con un porcentaje pequeño de espurios el cálculo de la solución es casi inmediato.

2.2.2. Método LMS

LMS proviene de las palabras inglesas Least Median of Squares (mínimo error de la mediana) [15]. Este método, al igual que el método RANSAC es un método probabilístico.

La idea del método es similar a la que emplea RANSAC; se selecciona aleatoriamente un número de subconjuntos y se calcula la homografía para cada uno de ellos. En este caso, en lugar de realizar votaciones, lo que se hace es calcular los residuos que genera el conjunto total de emparejamientos para cada homografía. De todas las soluciones obtenidas el método escoge aquella que minimice la mediana de los cuadrados de los residuos.

La ventaja de este método frente a RANSAC es que no necesita fijar umbrales. Sin embargo, es más costoso computacionalmente.

Partiendo de la hipótesis de que los datos se ajustan a una distribución normal, para eliminar los emparejamientos espurios el método LMS calcula un umbral σ en función del valor de la mínima mediana obtenida, y elimina todos aquellos emparejamientos cuyo residuo sea superior a dos veces el cuadrado de dicho valor.

$r_i^2 \leq (2 * \hat{\sigma}^2)$

2.2.3. Optimizaciones en los algoritmos

Para optimizar el cálculo de las homografías se puede utilizar la solución que se obtiene de los emparejadores robustos como solución definitiva o bien realizar una criba de espurios y resolver el sistema sobredimensionado para obtener una matriz de homografía. En el programa se ha incorporado la opción de realizar o no una criba de espurios y resolver el nuevo sistema sobredimensionado. En el capítulo de experimentos realizados y en el de resultados se trata con más detalle esta idea.

En el apéndice C se adjunta una información mucho más detallada acerca de las técnicas de emparejamiento robusto utilizadas.

2.3. Cálculo de homografías

En coordenadas homogéneas, para transformar puntos del plano se emplean matrices 3x3 denominadas *matrices de transformación*. La transformación de un punto **p** a otro **q** mediante una matriz de transformación tiene la forma:

$$\mathbf{p} = \mathbf{H}\mathbf{q}. \tag{2.2}$$

En esta parte del proceso se trata de encontrar una matriz **H** que satisfaga la condición 2.2 para los puntos **p** y **q** emparejados. Como no se puede satisfacer para todo el conjunto de emparejamientos a la vez, el objetivo es satisfacer 2.2 con el menor error posible, por ejemplo en el sentido de mínimos cuadrados.

Calcular una matriz de transformación 3x3, por lo tanto con 8 parámetros a determinar ya que existe un factor de escala, requiere 8 ecuaciones, y en consecuencia, se necesita un mínimo de 4 emparejamientos entre las características extraídas de las dos imágenes que se estén considerando. En las condiciones en las que se desarrolla este proyecto el movimiento va a ser siempre plano y las imágenes van a tener muy poco *baseline*, desplazamiento muy pequeño entre los centros ópticos. Partiendo de este hecho se puede realizar una simplificación en el problema eliminando de los cálculos la segunda coordenada de los puntos.

Con esta simplificación, el problema del cálculo de la matriz de transformación queda reducido al cálculo de una matriz **H** de dimensión 2x2, que cumpla, para los emparejamientos obtenidos

$$\begin{pmatrix} \lambda \mathbf{x}_1 \\ \lambda \end{pmatrix} = \begin{pmatrix} h_{11} & h_{12} \\ h_{21} & h_{22} \end{pmatrix} \begin{pmatrix} \mathbf{x}_2 \\ 1 \end{pmatrix} \tag{2.3}$$

donde $\mathbf{x}_1$ representa la coordenada x del punto $\mathbf{p}$, extraído de la primera imagen, y $\mathbf{x}_2$ representa la coordenada x del punto $\mathbf{q}$ (emparejamiento de $\mathbf{p}$) en la segunda imagen. En este caso solo hay que determinar 3 parámetros, cosa que se puede hacer utilizando únicamente 3 emparejamientos.

Si suponemos que todas las características extraídas se encuentran en un mismo plano, la matriz que se obtiene se conoce con el nombre de matriz de homografía. En [2] se demuestra que existe una relación entre dicha matriz y el ángulo de rotación que se ha producido en la cámara para pasar de una imagen a la otra siempre que el movimiento de traslación sea pequeño

El desarrollo de las ecuaciones para calcular la matriz de homografía a partir de los emparejamientos, así como una explicación más detallada de sus características, se encuentra detallado en el apéndice D.

Capítulo 3
Algoritmos de navegación

Se puede definir la navegación de un robot como el movimiento que sigue entre un punto inicial, que llamaremos punto de partida, y un punto final, denominado objetivo. Cuando el robot conoce las coordenadas de ambos puntos es capaz de desplazarse de uno a otro utilizando sus sensores odométricos. Sin embargo, la medida del desplazamiento mediante sensores de odometría es muy poco precisa y además los errores se van acumulando. Al final, el error total puede ser muy grande. Es por ello que, aunque de acuerdo con la odometría el robot haya llegado a su objetivo, este se encuentra en una posición completamente distinta.

Se ha dedicado mucho trabajo de investigación a la búsqueda de técnicas de corrección que permitan a los robots introducir correcciones en sus trayectorias con el objetivo de mejorar la navegación y conseguir movimientos más fiables.

Las técnicas basadas en visión que se van a utilizar en este proyecto funcionan de la siguiente manera. En una primera fase, denominada de fase de aprendizaje, se guía al robot describiendo el movimiento que se desea que repita posteriormente de manera autónoma y se van tomando imágenes de referencia, que servirán posteriormente para efectuar las correcciones cuando el robot repita el desplazamiento sin intervención humana. Esta navegación autónoma es la segunda fase del sistema, y se denomina fase de repetición.

Las cuestiones que surgen al utilizar este esquema son la determinación de los instantes más adecuados para tomar imágenes de referencia, cómo utilizar las imágenes de referencia para saber qué correcciones hay que hacer y cuándo y cómo hay que llevarlas a cabo. Todos estos aspectos se tratan a lo largo de este capítulo. En particular se describen los algoritmos que se han diseñado para realizar navegaciones robustas con el robot empleando, además de la odometría, los datos de corrección que proporciona la matriz de homografía.

3.1. Estimación del ángulo de desviación

Si se quiere corregir los errores en el movimiento del robot, es necesario conocer cuáles son estos errores. Como se ha visto en el capítulo anterior, se puede utilizar la matriz de homografía para realizar una estimación del error del robot en su orientación, comparando imágenes tomadas en tiempo real con imágenes de referencia almacenadas previamente. Dicha estimación proporciona el ángulo que se tiene que girar la cámara sobre su eje vertical para que la segunda imagen coincida con la primera, es decir, el ángulo que hay que girar el robot para corregir su desviación.

Para obtener el valor exacto de la corrección se ha empleado el parámetro h_{21} de la matriz de homografía. Después de varios experimentos, se ha ajustado el valor de la corrección, es decir, el ángulo de giro θ entre las imágenes, con la siguiente fórmula:

$$\theta = -\arctan(h_{21}) \tag{3.1}$$

En [2] se justifica la bondad en el uso de este parámetro, h_{21} como estimador del error de orientación. Los otros parámetros de la matriz también se pueden usar como estimadores pero presentan una mayor sensibilidad al ruido y a los errores en la matriz, por lo que se comportan de manera menos fiable.

Si tenemos una imagen que representa una referencia tomada previamente por el robot, entonces habrá que conseguir, mediante correcciones en su movimiento, principalmente rotaciones, que las imágenes que éste tome en tiempo real se aproximen todo lo posible a la de referencia.

3.2. Correcciones del ángulo de desviación

Notemos que no solo basta con conocer el ángulo de corrección necesario, sino que también hay que saber como utilizarlo en la navegación. Para realizar las correcciones en tiempo real hay que tener en cuenta el tipo de movimiento que está realizando el robot. Si el robot se está desplazando hacia adelante (o está retrocediendo), el programa de control deberá asignarle una velocidad angular dependiendo del valor obtenido en la comparación de imágenes. De esta forma el robot consigue volver a situarse en la recta del recorrido aprendido. La imagen 3.1 muestra un ejemplo de esta situación.

Cuando el robot se encuentra realizando un giro, realizar la corrección resulta más complicado. Por una parte, puesto que el robot está girando, la comparación de las imágenes necesariamente va a indicar que se debe realizar un giro. No hay ninguna forma de saber qué parte de este valor corresponde a corrección y qué parte corresponde al giro que se está realizando. Modificar la velocidad angular en tiempo real no aporta precisión en el giro. En capítulos posteriores se exponen algunas soluciones que se han probado, y los resultados que se han obtenido para resolver este problema con cada una de ellas.

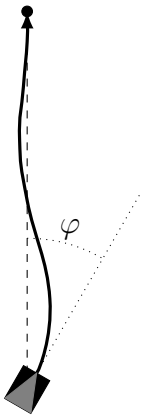

Figura 3.1: Ejemplo de corrección seguida por el robot en un desplazamiento longitudinal. La línea discontinua muestra el recorrido que debería seguir el robot y la linea de trazo continuo muestra el recorrido real seguido al comenzar con un error de orientación.

Es importante observar que en la mayoría de las situaciones, las correcciones necesarias son muy pequeñas, apenas unos grados. Para evitar que el robot realice giros demasiado bruscos serán necesarias velocidades angulares no mayores que unos pocos grados por segundo.

3.3. Algoritmos de navegación

Existen numerosos algoritmos de navegación basados en visión por computador. Algunos investigadores, como Chen [10] proponen mantener velocidad lineal constante y utilizar las imágenes para realizar las correcciones de rotación, mediante un sistema de votaciones sobre las características emparejadas. Otros algoritmos emplean exclusivamente la odometría para el movimiento, y utilizan las imágenes para realizar correcciones de rotación [1].

Para este proyecto se ha decidido utilizar una versión modificada del algoritmo de navegación que propone Yoshio Matsumoto [11] [12]. Este algoritmo propone añadir marcas de movimiento a las imágenes de referencia, de esta forma el robot sabe en cada momento el tipo de movimiento que debe realizar (avanzar, girar a la izquierda, a la derecha, etc...). Se ha elegido este algoritmo por varios motivos. En primer lugar, el uso de marcas de navegación en las imágenes de referencia permite realizar movimientos más complejos que los permitidos en [10]. La idea de este algoritmo se adapta muy bien a la corrección en tiempo real de la trayectoria utilizando matrices de homografía.

La versión que se ha implementado como parte de este proyecto, incorpora además información de la odometría del robot en el instante en que se ha tomado cada imagen. La combinación del uso de la odometría con los valores de las correcciones que se obtienen en tiempo real ha permitido conseguir unos resultados mucho mejores que los que se obtienen utilizando cada uno de los datos por separado.

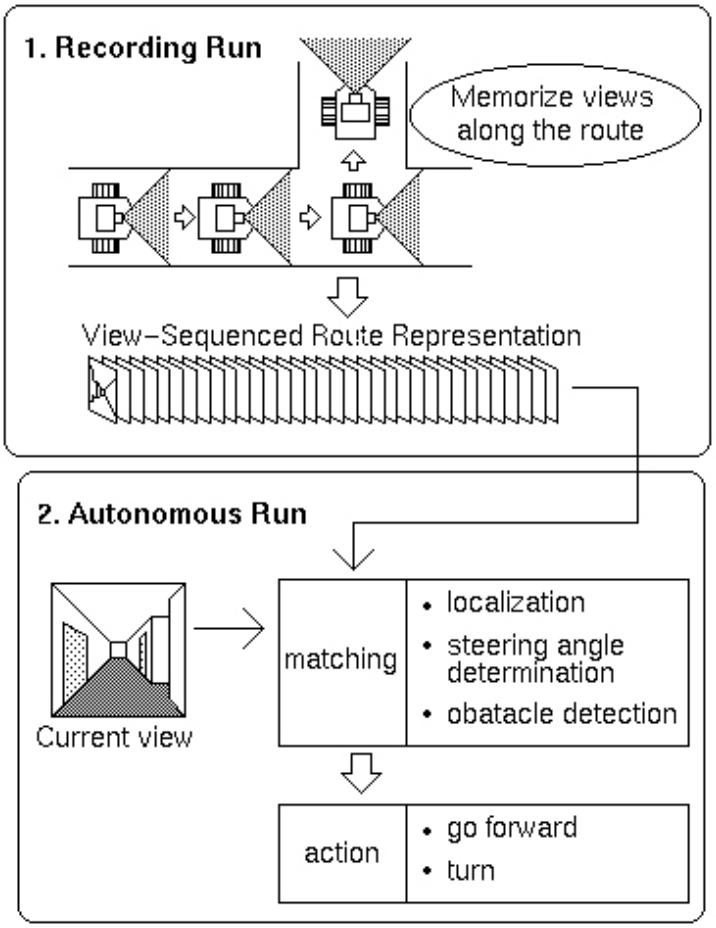

Figura 3.2: Esquema del algoritmo de navegación utilizado [11].

3.3.1. Fase de aprendizaje

En esta fase el robot capta imágenes y almacena sus características conforme efectúa el recorrido a repetir. Un punto importante es saber cuándo hay que almacenar la información de una imagen. Si se almacenan muchas imágenes, se pierde eficiencia; si, por el contrario no se toman suficientes, el control de correcciones en la fase de repetición no será fiable. Si se emplea la odometría como único criterio para determinar los instantes en los que se almacenan imágenes (por ejemplo, almacenar una imagen cada metro recorrido), podemos obtener resultados pobres porque no se tiene en cuenta la calidad de las características extraídas.

En el algoritmo diseñado se ha tenido en cuenta la odometría, y como segunda condición para el almacenamiento de nuevas imágenes de referencia la corrección respecto a la última imagen de referencia almacenada. Si el robot ha capturado una imagen en un instante del recorrido, conforme se vaya desplazando irá tomando imágenes y el valor de la

corrección que obtendrá para estas imágenes respecto a la de referencia irá aumentando. Cuando el valor de la corrección supere un determinado umbral, el robot almacenará la nueva imagen y la tomará como nueva referencia. De esta forma al acabar el recorrido, el robot ha almacenado un conjunto de imágenes que difieren entre ellas el umbral de corrección o se encuentran a una distancia determinada.

Es importante observar que cuando se habla de almacenar una imagen, no se almacena la imagen como tal, sino solamente las características que se han extraído de ella y la odometría del robot en la que se ha tomado. De esta forma se ahorra una gran cantidad de espacio en disco y la necesidad de volver a extraer las características de la imagen en la fase de repetición.

Ahora bien, si solo se almacenan las características extraídas de cada imagen, en la etapa de repetición el robot no sabrá qué clase de movimiento debe realizar para repetir la trayectoria. Para que esto no ocurra, a cada imagen se le asocia una etiqueta de movimiento que indique al robot lo que debe hacer. Las etiquetas que se han empleado son la siguientes:

- AVANZAR (indica que el robot debe moverse hacia adelante a partir de esta imagen)
- RETROCEDER (indica que el robot debe moverse hacia atrás a partir de esta imagen)
- GIRAR_IZQUIERDA (indica que el robot debe girar hacia la izquierda a partir de esta imagen)
- GIRAR_DERECHA (indica que el robot debe girar hacia la derecha a partir de esta imagen)
- PARAR (identifica la última imagen del recorrido, correspondiente al punto de destino)

Además de todo lo anterior, el robot también almacena la posición, indicada por la odometría, en que ha tomado la imagen. De esta forma, en la fase de repetición se podrá decidir mediante un criterio más robusto cuando se debe avanzar de imagen dentro de la lista obtenida durante el aprendizaje, como se explicará más adelante.

3.3.2. Fase de repetición

En la fase de repetición el robot utiliza las imágenes de referencia tomadas en el aprendizaje para repetir la trayectoria. Si el robot sólo tiene en consideración una imagen de referencia en cada momento, será capaz de corregir su error de orientación respecto a dicha imagen pero no sabrá en qué momento debe pasar a la siguiente de la lista. Para solventar esta dificultad, el robot trabaja en cada instante con dos imágenes de referencia consecutivas.

El proceso puede considerarse de la siguiente forma: partiendo de una posición correspondiente a una imagen de referencia, el robot debe desplazarse en el sentido indicado por la etiqueta de movimiento de dicha imagen hasta llegar a la posición asociada a la siguiente imagen de referencia. Inicialmente la corrección con respecto de la primera imagen será prácticamente nula, mientras que la corrección con respecto a la siguiente imagen de referencia será probablemente próxima al umbral de corrección fijado en la etapa de aprendizaje.

Es de esperar que conforme el robot se desplaza la corrección respecto de la primera imagen vaya aumentando, mientras que la corrección con respecto de la segunda irá disminuyendo. En la práctica estas variaciones tienen un comportamiento casi lineal, tal y como muestra la figura 3.3. Cuando la corrección respecto de la primera imagen sea suficientemente alta, y la corrección respecto de la segunda esté suficientemente próxima a cero, significará que se ha alcanzado la posición correspondiente a la segunda imagen y, por tanto, ha llegado el momento de avanzar en la lista de imágenes, repitiendo para las nuevas imágenes el proceso anterior.

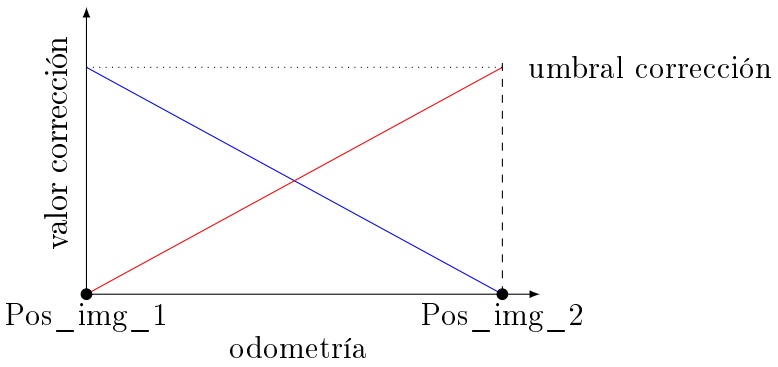

Figura 3.3: Evolución del valor de las correcciones. En rojo se observa el valor de la corrección respecto a la primera imagen y en azul el valor de la corrección respecto a la segunda.

Si sólo se emplean los valores de las dos correcciones para cambiar de imagen de referencia, puede ocurrir que el cambio se realice demasiado pronto o muy tarde. Entre imágenes que tienen la misma etiqueta de movimiento, estos desfases no representan un gran problema, ya que la antelación en un cambio se compensa con el retraso en el siguiente y viceversa. Sin embargo, cuando el cambio de imagen requiere de un cambio de tipo de movimiento, los errores que se produzcan pueden condicionar el resultado de la trayectoria. Un pequeño error en la orientación puede generar grandes errores en la posición si el robot se desplaza longitudinalmente.

También se ha visto que emplear la odometría como única medida para determinar el instante en el que hay que cambiar de imagen de referencia no es adecuado, ya que los errores en la odometría son acumulativos, y al final el robot nunca se encuentra donde se supone.

Inteligencia artificial para el avance de imágenes de referencia

El algoritmo de Matsumoto sólo emplea los valores de las correcciones obtenidas para realizar el avance en la lista de imágenes de referencia. Las primeras pruebas realizadas con este algoritmo mostraron que utilizar únicamente estos valores con las matrices de homografía no conseguía los resultados deseados.

Para mejorar el algoritmo se ha dotado al sistema de una inteligencia artificial que tenga en cuenta los valores de las correcciones y los valores de la odometría.

La IA evalúa un conjunto de funciones que comparan los valores de las correcciones entre sí, los valores de las correcciones con unos umbrales determinados empíricamente y los valores de la odometría del robot respecto a las posiciones de las imágenes de referencia. Cada función considerada devuelve un valor booleano que indica si la condición se cumple o no. En el momento en el que el número de condiciones supera un determinado umbral se considera que es el momento adecuado para avanzar en la lista de imágenes de referencia.

Llamando c1 a la corrección respecto a la primera imagen de referencia, c2 la corrección respecto a la segunda imagen y d la distancia que separa al robot de la posición donde se ha tomado la segunda imagen de referencia, las funciones de condición son:

- $|c1|-|c2| > 0$
- $||c1|-|c2|| >$ UMBRAL_DIFERENCIA
- $|c2| <$ UMBRAL_MINIMO
- $|c2| <$ UMBRAL_MINIMO y $|c1| >$ UMBRAL_MAXIMO
- $d <$ UMBRAL_DISTANCIA
- $d_i > d_{i-1}$

La solución mixta que se ha adoptado ha resultado ser muy ventajosa en la clase de trayectorias que se han considerado durante la fase de experimentación.

Reparaciones de la orientación del robot

Otra ampliación respecto al algoritmo de Matsumoto ha sido la realización de correcciones especiales en la orientación cada vez que el robot debía cambiar su tipo de movimiento; estas correcciones se han denominado reparaciones de orientación para diferenciarlas de las correcciones normales que realiza el robot durante su desplazamiento.

Partiendo de una imagen de referencia, una reparación de la orientación consiste en hacer que el robot gire lentamente hasta situarse orientado en la dirección de la imagen. El uso de velocidades de rotación pequeñas hace que la precisión del giro sea mucho mayor.

En la imagen 3.4 se muestra un esquema del funcionamiento de la reparación.

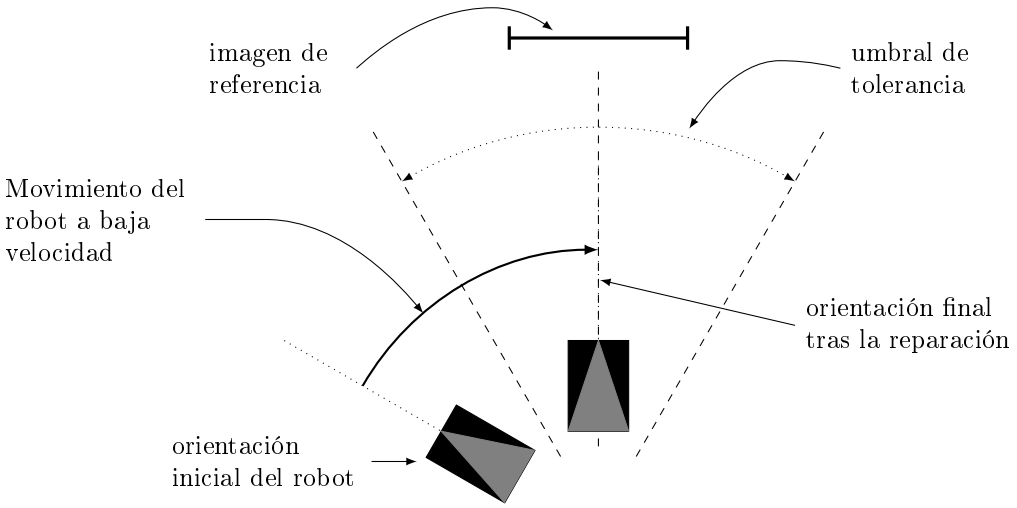

Figura 3.4: Reparación de la orientación. En la imagen se muestra la posición de partida del robot, la imagen de referencia, el error permitido y como, al final, el robot termina con la misma orientación que la imagen de referencia

Para que la reparación funcione correctamente, el robot debe estar situado (aproximadamente) sobre el eje de giro que lo lleve a la imagen de referencia y su error inicial de orientación no debe ser excesivamente grande. Estas condiciones se suelen dar siempre durante la realización de trayectorias en las que se han ido corrigiendo los errores de orientación con la matriz de homografía.

Estas reparaciones han permitido al robot comenzar cada movimiento situado en una orientación adecuada. Esta modificación ha resuelto los problemas que tenía el robot cuando debía desplazarse después de haber realizado un giro y viceversa.

3.4. Aplicación desarrollada

Al final del proyecto, todos los algoritmos que se han explicado en este capítulo se han integrado en una aplicación que permita a un robot *Pioneer* con un cámara digital integrada realizar una navegación robusta empleando la visión por computador.

En esta memoria se ha adjuntado un manual de usuario que explica detalladamente todas las opciones que incorpora la aplicación desarrollada. Esta información se encuentra en el apéndice E.

Si en algún momento se decide ampliar esta aplicación dotándola de un mayor número de extractores de características o técnicas de emparejamiento robustas, para simplificar

la tarea de los desarrolladores, se ha incluido un apéndice en el que se detalla toda la estructura interna de la aplicación. En el apéndice se incluye la gestión de configuraciones, la organización de los módulos y un listado completo de los ficheros de código. Con esta información se pretende ahorrar tiempo a futuros desarrolladores, evitándoles tener que descifrar toda la estructura del software. Toda la información relacionada con la estructura interna del software se encuentra en el apéndice F.

Capítulo 4
Experimentos realizados

Una vez realizado todo el análisis teórico sobre las correcciones en tiempo real mediante visión, se han realizado varios experimentos que han permitido validar dicho análisis en todas las situaciones posibles.

Se han dividido los experimentos en varios bloques:

1. *Experimentos estáticos*: En estos experimentos se ha trabajado con datos generados aleatoriamente para simulación y con imágenes tomadas con el robot de forma estática. Se han empleado para comparar los diferentes métodos de extracción de características, técnicas de emparejamiento y para validar los cálculos de la matriz de homografía.

2. *Movimientos sencillos*: Para probar que el robot es capaz de realizar correcciones en su trayectoria debidas a errores de rotación se han realizado experimentos que impliquen movimientos sencillos del robot; rotaciones puras y desplazamientos. Estos experimentos han servido además para ajustar todos los parámetros que utiliza el sistema de visión.

3. *Trayectorias completas*: Se han realizado experimentos de mayor envergadura enseñando al robot trayectorias que incluyan distintos desplazamientos y rotaciones combinados en un mismo recorrido. De esta forma se ha comprobado como afecta el error acumulado en la navegación.

4.1. Experimentos estáticos

Una vez estudiado y analizado el código disponible al inicio del proyecto, se ha procedido a estructurarlo, modularizarlo e incorporarle nuevos módulos que doten al sistema de posibilidades adicionales. Se ha preparado un conjunto de experimentos que sirvan para validar cada uno de los módulos del sistema, verificar el código implementado y evaluar las prestaciones de los métodos planteados.

Inicialmente se han realizado experimentos utilizando datos generados aleatoriamente para comparar los distintos métodos de emparejamiento utilizados. Con los datos aleatorios se ha comprobado la tolerancia a datos espurios de los diferentes métodos, sin que se vean afectados por la extracción de las características en imágenes. Posteriormente, una vez finalizados los experimentos relacionados con la extracción de características, se han utilizado imágenes reales tomadas con el robot para medir tiempos de cómputo y verificar con datos reales la calidad de los emparejadores.

El segundo bloque de experimentos estáticos se ha orientado a la calibración de los parámetros de los extractores de características de las imágenes. Estos experimentos también han servido para deducir el tamaño adecuado de las imágenes capturadas. El objetivo ha sido alcanzar un equilibrio entre el tiempo de cómputo de los extractores y la cantidad y calidad de las características extraídas. Este segundo conjunto de experimentos ha servido para poner de manifiesto el problema del entrelazado cuando las imágenes se capturan en movimiento.

4.2. Movimientos sencillos

Las traslaciones o desplazamientos longitudinales suelen generar pocos errores de rotación. Sin embargo, si inicialmente existe un pequeño error de rotación, el desplazamiento longitudinal del robot generará un error de desplazamiento horizontal directamente proporcional al ángulo de desviación. En la figura 4.1 se observa claramente este fenómeno. Si el robot se una distancia x y hay un error de orientación de φ radianes, al acabar el desplazamiento habremos acumulado un error en el eje X de $x{\cdot}(1 - cos(\varphi))$ y habremos realizado un desplazamiento lateral de $x{\cdot}sin(\varphi)$ metros.

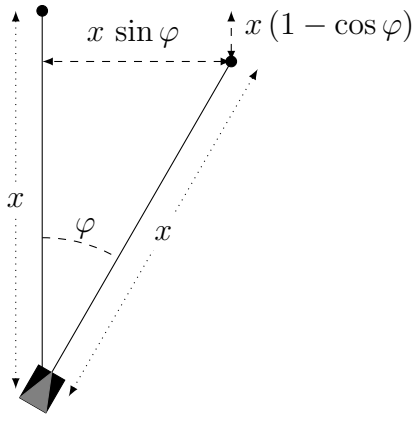

Figura 4.1: Ejemplo de errores cometidos por la orientación.

Partiendo de un desplazamiento longitudinal puro, se han realizado varios experimentos con el robot iniciando su camino con distintos ángulos de desviación. De esta forma se ha comprobado hasta que punto el sistema es capaz de corregir errores de rotación en

los desplazamientos.

La corrección de errores de rotación cuando el robot está girando es más compleja que la corrección durante un traslación. Una vez que se ha verificado el funcionamiento de la función encargada de reparar giros, se ha experimentado con varias soluciones al problema de los giros. En el capítulo de resultados se explican todas las soluciones probadas y los resultados obtenidos con cada una.

4.3. Trayectorias complejas

En último lugar se han realizado experimentos de mayor envergadura para ver hasta qué punto los métodos utilizados sirven para corregir trayectorias. Se han diseñado diversos tipos de trayectorias que incluyen tanto desplazamientos longitudinales como rotacionales. De esta forma se ha conseguido observar como afecta el tiempo y el tamaño del recorrido al error final cometido por el robot.

Estos experimentos también se han empleado para analizar la repercusión del umbral de corrección en la etapa de aprendizaje. Un umbral demasiado pequeño generará demasiada información mientras que un umbral demasiado alto almacenará muy poca. Se ha estudiado cuál es el valor más adecuado para que no genere información redundante ni permita que el robot pierda el rumbo.

Otra de las variables que se pretende estudiar en estos experimentos es la velocidad del robot, tanto en la etapa de aprendizaje como en la de repetición. Si en el aprendizaje la velocidad es demasiado alta al robot no le dará tiempo de procesar la información que obtiene del medio. Sin embargo, uno de los objetivos de la navegación es que el robot sea capaz de realizar las trayectorias lo antes posible.

Como los métodos utilizados para la navegación extraen información de las imágenes que toman, otro factor que se ha tenido en cuenta en estos experimentos ha sido el grado de preparación del entorno. En el caso de la extracción de contornos verticales se ha estudiado como afecta al sistema que el entorno tenga más o menos contornos disponibles para su detección. En el caso del extractor de características SURF se ha realizado algo similar teniendo en cuenta en qué lugares se extraen las características SURF.

Capítulo 5
Resultados obtenidos

En relación con los experimentos detallados en el capítulo 4, se resumen en este capítulo los resultados más destacados.

5.1. Calidad de los emparejadores

En primer lugar se ha realizado una simulación con datos aleatorios para estudiar la probabilidad con la que cada emparejador es capaz de generar la solución correcta (matriz de homografía) en función del porcentaje de espurios que haya en el conjunto de emparejamientos. Entendamos una prueba como la generación de una matriz de homografía $\mathbf{H}$ aleatoria y 35 puntos $\mathbf{p}_i$ también aleatorios. Con estos datos se han generado los 35 puntos $\mathbf{q}_i$ emparejados realizando la transformación:

$$\mathbf{q}_i = \mathbf{H}\mathbf{p}_i \tag{5.1}$$

de esta forma se garantiza que la matriz de homografía que relaciona los emparejamientos es la generada aleatoriamente. Se ha fijado un valor del tanto por ciento de espurios en la muestra y se distorsiona una parte del conjunto de puntos $\mathbf{q}$ de manera que el conjunto de emparejamientos resultante tenga el porcentaje de espurios deseado. Con estos datos se han calculado las matrices de homografía con los distintos emparejadores. Para decidir si una prueba ha finalizado con éxito se ha comparado el valor de la homografía calculada con el valor de la homografía original; si la diferencia entre ambas es inferior a un umbral determinado, se considera que el emparejador ha finalizado con éxito el cálculo. En caso de que el error supere el umbral o no haya sido posible el cálculo de la homografía, se considera que el método ha fracasado.

Para cada valor entero del porcentaje de espurios entre 1 y 100 se han realizado 1000 pruebas (1000 matrices de homografía diferentes) con cada emparejador. Se ha contabilizado el número de éxitos obtenido en cada caso, lo que nos ha permitido calcular el porcentaje de éxito de cada emparejador en función de la fracción de espurios de la muestra. También se ha calculado el error medio cometido en función de la fracción de espurios.

Las gráficas 5.1, 5.2 y 5.3 muestran los resultados obtenidos en el experimento.

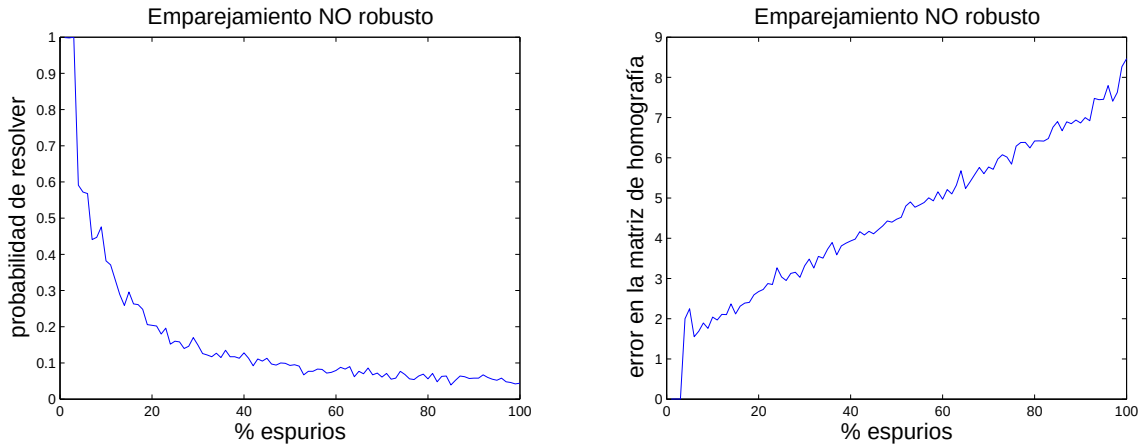

Figura 5.1: Resultados obtenidos empleando únicamente el emparejamiento básico de características. Para cada porcentaje de espurios se han realizado 1000 pruebas con diferentes matrices de homografía generadas aleatoriamente.

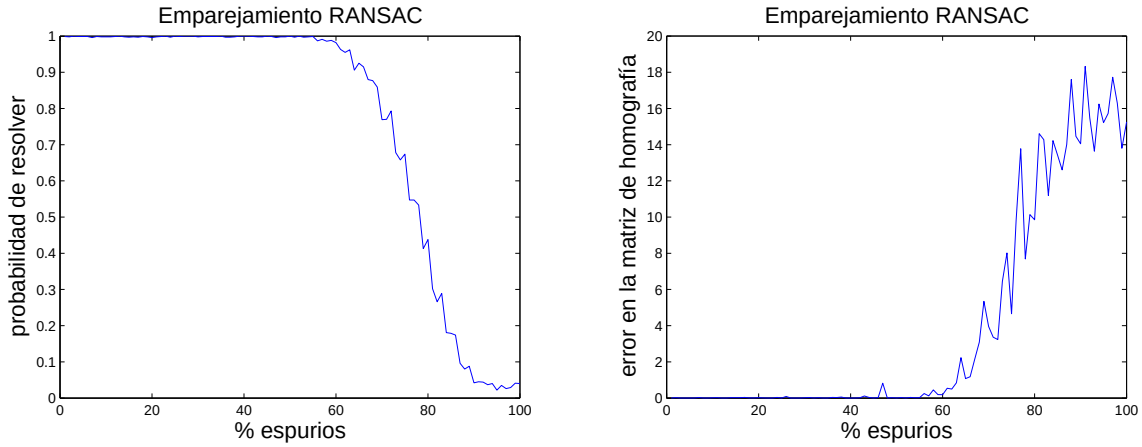

Figura 5.2: Resultados obtenidos utilizando un emparejamiento robusto con el algoritmo RANSAC. El número de pruebas realizado ha sido el mismo que el empleado con el emparejador básico.

Como se aprecia en las gráficas, el método de RANSAC es el más tolerante a espurios de emparejamientos, en el sentido de que tiene probabilidades más altas de generar soluciones exactas que el emparejador básico o el emparejamiento robusto utilizando LMS. Además los errores que se cometen en la matriz de homografía son en general menores que los que se cometen el emparejador LMS o los obtenidos resolviendo el sistema sobredimensionado.

Aunque el emparejamiento no robusto es peor que los emparejadores robustos, si el número de espurios es muy elevado, los errores que se cometen son menores. Esto es debido a que cuando hay más espurios que emparejamientos buenos, estadísticamente los errores positivos de unos emparejamientos malos se compensan con los errores negativos de otros,

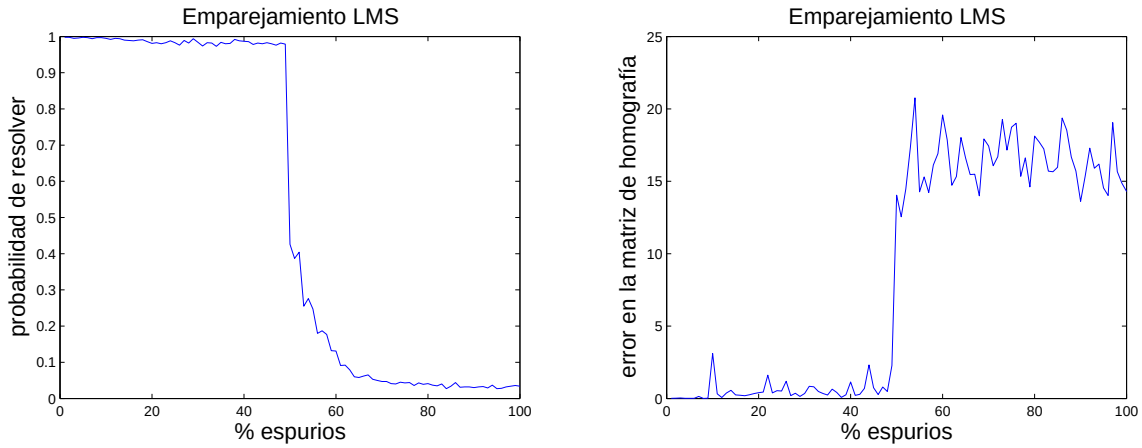

Figura 5.3: Resultados obtenidos un emparejamiento robusto con el algoritmo LMS. El número de pruebas realizado ha sido el mismo que el empleado con el emparejador básico.

y en promedio, la matriz resultante no tiene errores elevados. En el caso de RANSAC o LMS, al haber una fracción de espurios superior al conjunto de buenos emparejamientos es bastante probable que una mala solución reciba más votos que la solución buena, con lo cual, la matriz obtenida tiene poca relación con la matriz real, y el error resultante es mucho mayor.

5.2. Calidad de los sistemas de extracción de características

Comparar dos sistemas de extracción de características es una tarea compleja, ya que no existen criterios objetivos que se adapten a todas las situaciones. Cambiar el tamaño de las imágenes muestreadas o el contenido de las mismas puede hacer que los métodos parezcan mejores o peores. Para poder realizar una valoración más o menos consistente se han realizado diferentes pruebas teniendo en cuenta:

- Tamaño de las imágenes.
- Grado de acondicionamiento del entorno
- Ángulo de separación entre las imágenes.
- Consideración de imágenes en movimiento.

Las diferentes pruebas realizadas han ido manifestando las distintas ventajas e inconvenientes de los dos métodos (SURF y contornos verticales). Estas pruebas han servido también para encontrar y resolver problemas ajenos a los extractores como el problema del entrelazado.

5.2.1. Problema del entrelazado

Cuando las imágenes son tomadas durante el movimiento del robot aparece el problema del entrelazado. Como la cámara realiza muestreos de filas pares e impares de la imagen por separado, en los giros, las imágenes obtenidas aparecen distorsionadas (Figura 5.4). Esto afecta negativamente a ambos extractores, y en especial al de contornos verticales ya que en estas imágenes las líneas verticales aparecen con trazo en forma de zigzag y el extractor no es capaz de identificarlas. Un ejemplo del efecto producido por el entrelazado se puede ver en la figura 5.5.

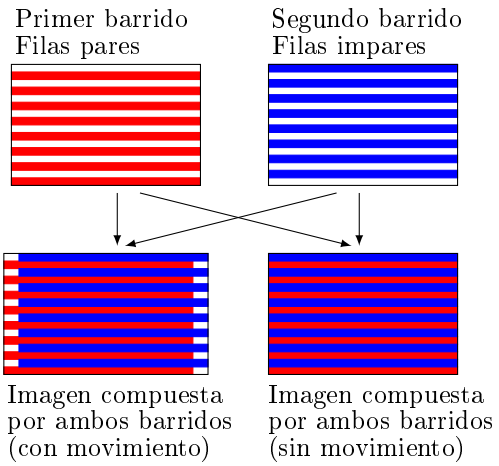

Figura 5.4: Muestreo de una imagen.

La solución que se ha adoptado para resolver este problema ha sido eliminar en las imágenes las filas pares. De esta forma no solo desaparece el problema del entrelazado, sino que además el tiempo de cómputo se reduce considerablemente al trabajar con imágenes la mitad de grandes. Es importante aclarar que la eliminación de la mitad de las filas, además, no ha afectado negativamente a los extractores.

Después de haber experimentado con varios tamaños de imagen, se ha visto que la resolución de 625x240 es la que mejores resultados ha proporcionado.

5.2.2. Tiempos de cómputo

El extractor de puntos SURF tiene un coste computacional claramente superior al del extractor de contornos verticales. Para que su uso sea efectivo en la práctica se han realizado algunas modificaciones en el algoritmo, orientadas a la reducción de los tiempos de cálculo.

Por una parte se ha conseguido agilizar mucho los cálculos del extractor SURF eliminando la invariancia a rotaciones. Como el robot se mueve sobre un mismo plano todo

Figura 5.5: Problema del entrelazado.

el tiempo los puntos extraídos no presentan rotaciones en la imagen. La segunda modificación ha consistido en reducir el tamaño del descriptor de cada elemento SURF de 64 a 36 elementos. Las pruebas realizadas han demostrado que en el entorno en que se está trabajando, la pérdida de precisión en el descriptor no afecta de manera negativa a la navegación. La utilización de técnicas robustas de emparejamiento compensa con creces esta reducción.

Con estas dos modificaciones se ha conseguido reducir el tiempo medio de cálculo del extractor, en el computador del robot, de aproximadamente 2 segundos a 300-400 milisegundos, para el tamaño de imagen citado en la subsección anterior, dependiendo del número de puntos que encuentre el extractor.

Aun habiendo reducido considerablemente el tiempo de cómputo del extractor SURF, este necesita que el robot se mueva a velocidades inferiores a las que permite el extractor de Burns para funcionar correctamente.

5.2.3. Homografías obtenidas

El hecho de que el extractor de características SURF sea más lento no lo convierte en peor extractor. La siguiente etapa de experimentos ha consistido en observar la calidad

de las homografías obtenidas para ambos métodos con diferentes pares de imágenes.

El extractor SURF ha resultado ser mucho más robusto en el cálculo de homografías. En primer lugar, los emparejamientos son más fiables, ya que se basan en descriptores de 36 elementos frente a las 6 parámetros que emplea el emparejador de rectas. Esto hace que el número de espurios sea mucho menor. Además, el número de puntos que se pueden extraer de una imagen es mucho mayor que el número de rectas que tienen una orientación casi vertical. En definitiva hay un mayor número de emparejamientos con un porcentaje menor de espurios, lo que se traduce en mayor fiabilidad en el cálculo de la homografía.

Con los experimentos se ha visto que las características SURF, para correcciones de hasta 0.7 radianes, tienen una probabilidad de éxito en el cálculo de casi el 100 % y la precisión de la solución es del orden de las milésimas. El extractor de contornos verticales, en cambio, tiene probabilidades de éxito inferiores y es muy dependiente de la imagen, por lo que para garantizar su correcto funcionamiento hay que acondicionar el entorno para que haya gran cantidad de rectas verticales.

5.3. Movimientos sencillos

5.3.1. Reparación de la orientación respecto a una imagen de referencia

Se han realizado varios experimentos para probar la función que se ha implementado para reparar la orientación del robot respecto a imágenes de referencia. En los experimentos se ha tenido en cuenta:

- El tipo de extractor empleado
- El giro a realizar
- Acondicionamiento del entorno
- Velocidad de giro
- Tolerancia (error permitido en la corrección)

Tomando una imagen de referencia se ha querido comprobar la calidad de la reparación de la orientación haciendo pruebas con distintos ángulos de error. Los resultados han demostrado que en un intervalo de ± 0.4 radianes, con una precisión menor a la tolerancia fijada, el robot es capaz de reparar el error la mayoría de las veces. Para valores que oscilen entre 0.4 y 0.7 la reparación depende casi totalmente del grado de acondicionamiento de las imágenes capturadas. Para valores superiores no se garantiza la posibilidad de llevar a cabo la reparación de manera correcta.

Se ha visto que existe una relación entre la velocidad que se emplea en el giro y la tolerancia que se debe utilizar para determinar que la reparación ha terminado correctamente. Esta relación es diferente para el extractor de contornos verticales que para el extractor de características SURF, ya que depende del tiempo de extracción.

Notemos que mientras se está calculando el factor de corrección de una imagen el robot sigue en movimiento. Por lo tanto, en el instante en que detectemos que la corrección respecto a una imagen es inferior a la tolerancia, el robot se habrá desplazado un poco respecto a la posición en que se tomó dicha imagen (véase imagen 5.6). Este desplazamiento depende de la velocidad de rotación del robot y del tiempo empleado para el cálculo de la corrección. Si la velocidad de giro es alta, hay que aumentar el valor de la tolerancia (tolerancia menos precisa) para que compense el exceso de desplazamiento cometido durante los cálculos. Por el contrario, si la velocidad es baja, habrá que usar valores más pequeños de la tolerancia (tolerancia más precisa).

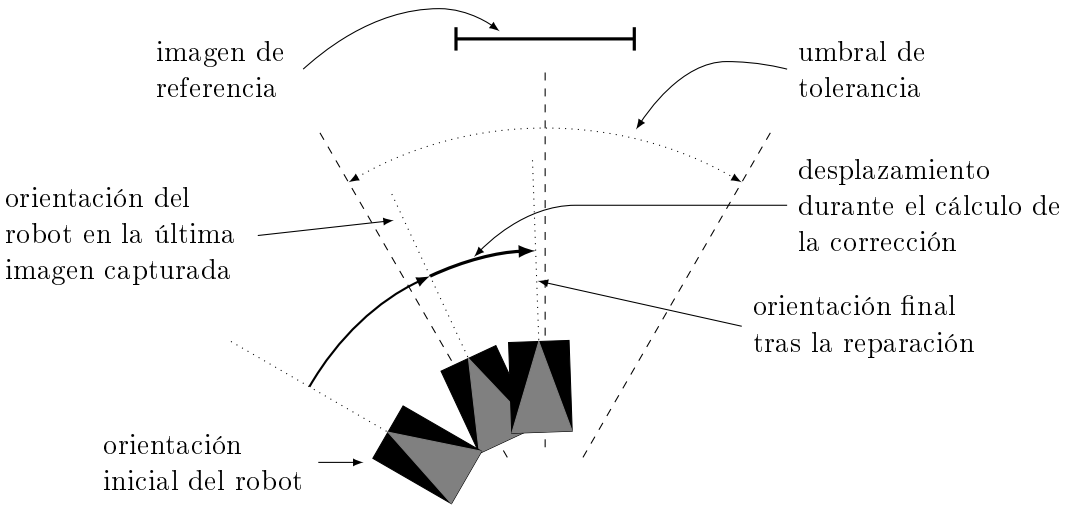

Figura 5.6: Reparación de la orientación. Partiendo de una posición en la que el robot se encuentra mal orientado, el robot realiza un giro hasta que detecte que la corrección de la última imagen capturada respecto de la imagen de referencia es inferior al umbral de tolerancia admitido. La posición final del robot no coincide con la posición en la que se ha capturado la última imagen ya que durante el tiempo empleado en el cálculo de la corrección el robot ha seguido moviéndose. Eligiendo adecuadamente la velocidad de giro y el umbral de tolerancia para que tengan en consideración el desplazamiento final se obtiene el resultado deseado: que el robot finalice su movimiento con la orientación adecuada.

Si el error tolerado es alto y la velocidad es pequeña ocurrirá que el robot no llegará al punto deseado de la reparación, aunque estará dentro de los márgenes admitidos (imagen 5.7). Por otro lado, si el error que se tolera es demasiado bajo en relación a la velocidad, el robot se pasará el destino a causa del desplazamiento ocurrido durante los cálculos (imagen 5.8).

La tabla 5.1 muestra la relación entre la velocidad de rotación y la tolerancia recomendada para los dos extractores utilizados.

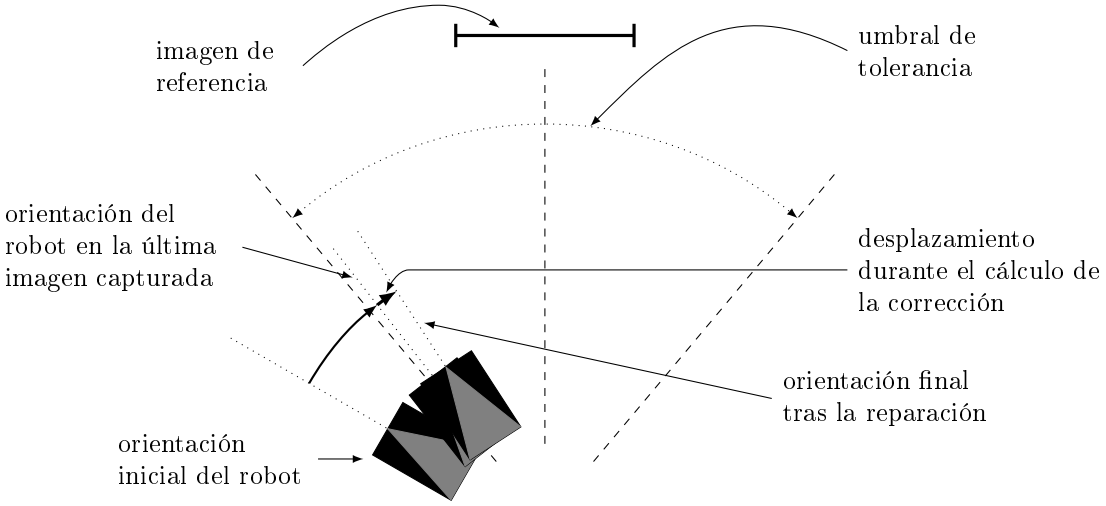

Figura 5.7: Reparación con velocidad lenta y margen de error grande. En este caso aunque el robot termina con la orientación dentro de los márgenes permitidos, se observa que la orientación final no es la correcta. Un umbral de error permitido alto, combinado con una velocidad de movimiento baja hacen que el robot no complete la reparación correctamente. Para resolver este error se tendría que reducir el umbral de error permitido.

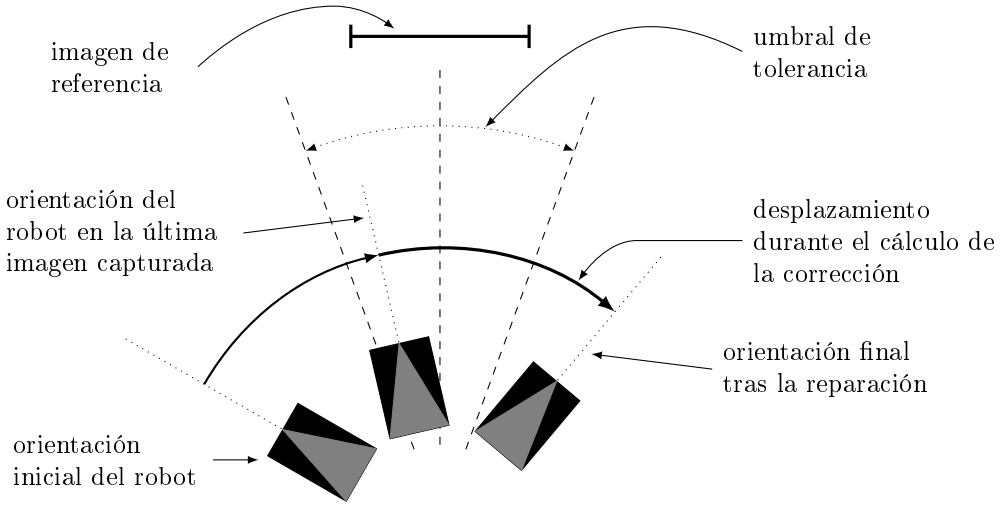

Figura 5.8: Reparación con velocidad alta y margen de error pequeño. En este caso el robot termina mal orientado porque realiza un desplazamiento muy grande desde que captura una imagen dentro del intervalo permitido hasta que obtiene el valor de corrección de dicha imagen. Para solucionar este problema habría que reducir la velocidad de movimiento del robot.

También se deduce de estos experimentos que para velocidades muy altas disminuir el error tolerado no va a generar movimientos más precisos y no se puede garantizar una reparación de calidad.

Contornos verticales		Puntos SURF	
Velocidad (rad/s)	Tolerancia (rad)	Velocidad (rad/s)	Tolerancia (rad)
0.01	0.04	0.01	0.04
0.02	0.04	0.02	0.05
0.03	0.08	0.03	0.1
0.04	0.1	0.04	0.12

Tabla 5.1: Relación entre la velocidad de giro y la tolerancia en la reparación.

5.3.2. Correcciones en desplazamientos longitudinales

Para los experimentos en desplazamientos longitudinales se ha considerado una trayectoria únicamente rectilínea durante el aprendizaje. En la posterior fase de repetición se ha colocado al robot con diferentes errores de rotación iniciales para observar su comportamiento.Los resultados han presentado sensibilidad a dos factores distintos.

Por un lado, como es de esperar, el ángulo de error inicial es un factor que afecta al resultado. Si el error inicial es superior a ± 0.8 radianes se ha visto que el robot no es capaz de recuperarse, independientemente del tipo de extractor utilizado.

Para errores inferiores, aparte de la dependencia de las extracciones al condicionamiento del entorno, el segundo factor que se ha considerado es la longitud del desplazamiento. Si la longitud es grande el robot dispone de más tiempo para corregir su trayectoria, y por tanto es capaz de corregir errores mayores. Si, por el contrario, se dispone de poco espacio para la corrección, aunque el robot fuera capaz de corregir el ángulo de desviación, se detendrá antes de que le de tiempo.

Estos resultados hacen que sea de la máxima importancia que el robot empiece todos sus desplazamientos con la mejor orientación posible.

5.3.3. Correcciones en giros

La corrección de los giros es, posiblemente, el problema más complicado de resolver que ha aparecido durante la realización de este proyecto. Se han realizado experimentos con cuatro tipos de soluciones al problema para estudiar la viabilidad de cada una de ellas.

No corregir durante el giro

La primera solución probada ha sido no realizar reparaciones de orientación durante el giro y dejar que sea en los posteriores desplazamientos longitudinales donde se realice la corrección.

Como se ha visto en la subsección anterior, la corrección de un error inicial de rotación

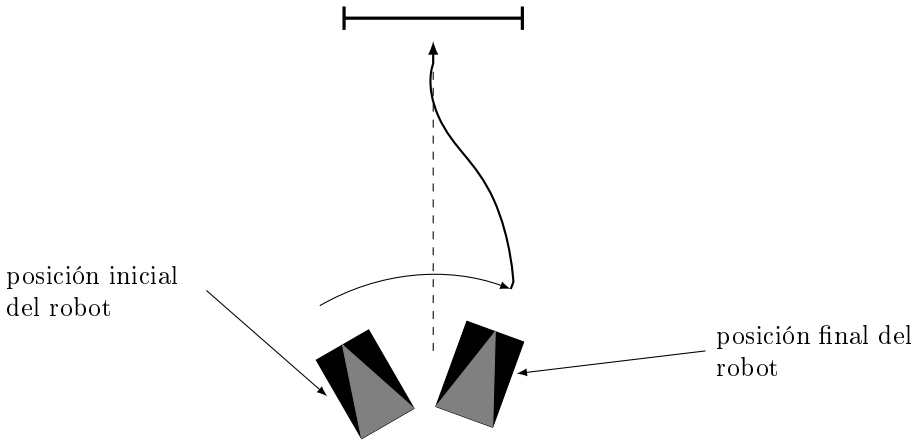

Figura 5.9: Ejemplo de giro sin realizar reparaciones. El robot realiza un giro utilizando la odometría cometiendo un error. La corrección se realiza en el posterior desplazamiento longitudinal.

a lo largo de un desplazamiento longitudinal, depende principalmente del error a corregir y de la distancia que haya para corregirlo. En una rotación, excepto en casos excepcionales, los errores que se cometen oscilan entre -0.2 y 0.2 radianes. Se ha visto que para este caso realizar la corrección en el desplazamiento es posible.

Sin embargo, está solución se ha descartado, principalmente por dos motivos. El primero de ellos es el desconocimiento de la distancia que hay para corregir posteriormente el error; se ha procurado que cada fase del movimiento fuera lo más precisa posible y esta solución funciona justamente al revés. El segundo motivo viene cuando el último movimiento de la trayectoria es una rotación; en este caso no habría posibilidad de corregir más adelante el error y el robot no podría terminar de forma correcta el recorrido.

Realizar una reparación por cada imagen de referencia

Esta solución, de todas las utilizadas, es la que ha presentado peores resultados en los experimentos. Puesto que la distancia entre imágenes de referencia oscila entre 0.05 y 0.3 radianes la realización del giro a velocidad normal es muy imprecisa en proporción al giro realizado (entre un 50 % y un 200 %). Generalmente este error en el giro, además, es por exceso; esto implica que el robot esté girando constantemente hacia la izquierda y hacia la derecha para reparar el exceso de giro. Esto provoca grandes errores en la odometría del robot.

Por todo ello se ha desechado el uso de esta solución en la realización de trayectorias complejas.

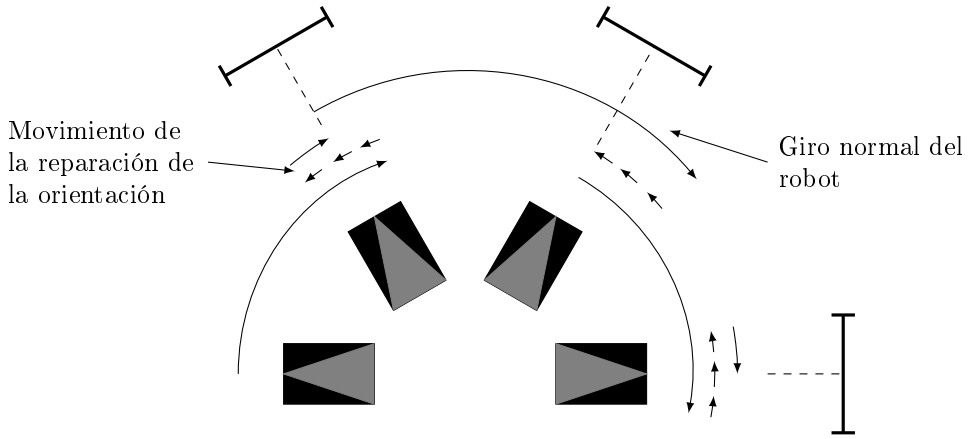

Figura 5.10: Ejemplo de giro realizando una reparación en cada imagen de referencia. El robot gira usando la odometría hasta la posición de cada imagen de referencia. Puesto que el giro no es perfecto realiza una reparación respecto a dicha imagen de referencia para quedar bien orientado. En este caso se observa que el robot está cambiando el sentido de sus giros constantemente; esto, además de ser ineficiente, provoca errores mayores en la odometría.

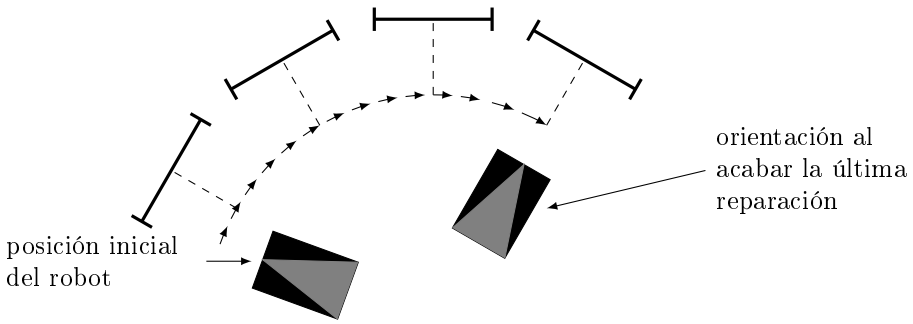

Figura 5.11: Ejemplo de giro realizado mediante reparaciones. El robot realiza el giro como una sucesión de reparaciones de orientación respecto a cada una de las imágenes de referencia. Se observa la precisión del giro, sin embargo, el tiempo empleado es mucho mayor que para las otras soluciones.

Realizar el giro como una sucesión de reparaciones

La idea para este caso consiste en realizar todo el giro como una sucesión de reparaciones respecto a las imágenes de referencia tomadas. Puesto que en todos los experimentos el umbral de corrección ha sido inferior a 0.3 radianes, el robot ha sido perfectamente capaz de realizar los giros con esta solución, obteniendo además una gran precisión.

El problema de esta solución es su ineficiencia. El robot emplea en realizar la reparación respecto a una imagen el mismo tiempo que le costaría realizar todo el giro completo sin correcciones. Si el giro que debe realizar es relativamente grande (del orden de $\pi/2$), el tiempo invertido es demasiado alto para poder considerar la viabilidad de la solución.

Realizar una reparación al terminar el giro

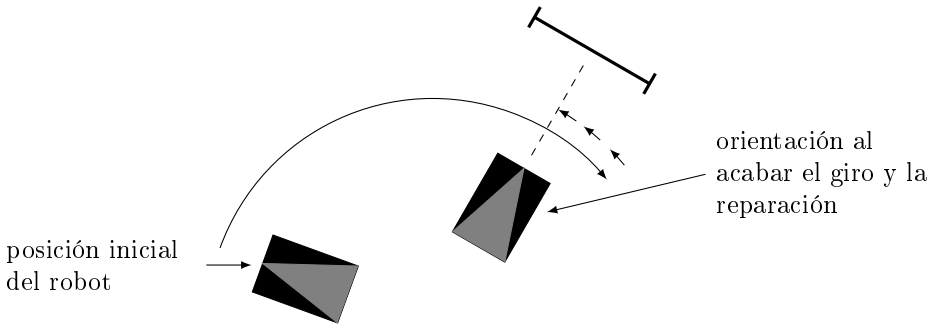

Figura 5.12: Ejemplo de giro realizando una reparación al finalizar la rotación. El robot realiza el giro completo sin hacer reparaciones. Al finalizarlo realiza una reparación respecto a la última imagen de referencia del giro. De esta forma se garantiza que el robot comienza el siguiente movimiento con la orientación deseada.

La última solución probada ha consistido en realizar el giro completo sin correcciones y, antes de empezar el siguiente desplazamiento, realizar una reparación que garantice que el robot empieza a moverse con una buena orientación.

Esta solución ha sido la que mejores resultados ha dado. En primer lugar, el tiempo invertido en el giro es mínimo, ya que lo realiza a velocidad de rotación normal. Además se garantiza que el siguiente movimiento comienza correctamente ya que se ha reparado al final el posible error cometido.

Una ventaja adicional de esta solución ha sido la eliminación de las imágenes de referencia intermedias tomadas durante los giros. Conociendo el valor del giro a realizar no es necesario que haya imágenes intermedias. Teniendo la imagen de referencia al finalizar el giro, el robot es capaz de reparar su orientación hasta colocarse correctamente.

Con esta solución no solo se obtienen mejores resultados, sino que se reduce la cantidad de información que es necesario almacenar.

5.4. Trayectorias complejas

La última fase de experimentos ha consistido en la realización de trayectorias más complejas que contuvieran una combinación de los movimientos de giro y rectilíneos.

Estos experimentos han puesto de manifiesto las ventajas del uso de la visión por computador, aunque también han servido para mostrar algunas carencias de los métodos de extracción de características.

5.4.1. Umbral de corrección

Una buena elección del umbral de corrección en la etapa de aprendizaje ha demostrado que afecta positivamente a los resultados obtenidos durante la fase de repetición. Puesto que en las rotaciones solo son necesarias dos imágenes de referencia (una al comienzo y otra al final), la elección del umbral de corrección debe orientarse a la obtención de un número adecuado de imágenes durante los desplazamientos longitudinales.

Tras un buen número de experimentos en diversas condiciones, se ha concluido que un umbral que resulta ser muy bueno en estas situaciones para cualquier velocidad en el aprendizaje (dentro del intervalo de velocidades [0.1-0.5] m/s y rad/s) y cualquiera de los dos extractores de características ha sido el de 0.1 radianes en la corrección. Con este umbral se almacena una imagen cada metro aproximadamente. La elección de un valor menor provoca un exceso de información que, aunque no perjudica la navegación, tampoco la mejora. Un valor superior a 0.1 hace que la distancia entre las imágenes sea mayor y los valores de las correcciones resulten inestables.

5.4.2. Velocidades de movimiento

Utilizar una velocidad más pequeña en la fase de repetición que en la fase de aprendizaje ha proporcionado mejores resultados en los experimentos. Esto tiene bastante sentido, ya que en la etapa de repetición se realizan dos cálculos de correcciones en cada iteración del algoritmo mientras que en la etapa de aprendizaje solo es necesario un cálculo por iteración.

El menor tiempo de cómputo del extractor de contornos verticales ha permitido una mayor velocidad en ambas fases de la navegación. En la tabla 5.2 se han mostrado las velocidades que han dado mejores resultados con ambos extractores.

	Contornos verticales		**Puntos SURF**	
	Aprendizaje	Repetición	Aprendizaje	Repetición
Movimiento lineal	0.2 m/s	0.12 m/s	0.15 m/s	0.08 m/s
Movimiento rotacional	0.2 rad/s	0.1 rad/s	0.15 rad/s	0.06 rad/s

Tabla 5.2: Velocidad máxima en la fase de aprendizaje y en la de repetición para que el robot navegue correctamente.

5.4.3. Acondicionamiento del entorno

El grado de preparación de los escenarios ha sido otro factor importante a tener en cuenta.

La utilización del extractor de contornos verticales ha implicado una necesidad mayor de preparar los escenarios para garantizar el número de características extraídas.

Con SURF, al extraerse un mayor número de puntos por imagen, y además con mayor calidad, no ha sido necesario preparar tanto el entorno; sin embargo, la colocación de algunos elementos, como tablones con imágenes, que ayuden al extractor hace que el movimiento del robot sea mejor.

5.4.4. Imágenes tomadas durante el aprendizaje

A continuación se muestra un conjunto de imágenes capturadas durante el aprendizaje de una trayectoria por el laboratorio de robótica.

Figura 5.13: Imágenes capturadas durante un aprendizaje empleando extracción de características SURF (ordenadas de izquierda a derecha y de arriba a abajo).

Capítulo 6
Conclusiones y trabajos futuros

6.1. Conclusiones

Después de todos los experimentos realizados, y a la vista de los resultados obtenidos, se pueden extraer las siguientes conclusiones:

La navegación autónoma de robots es imprecisa y supone un problema complicado de resolver. La calidad de los sensores que incorporan los robots resulta ineficiente para controlar adecuadamente el robot en las aplicaciones habituales. Sin embargo, teniendo en cuenta que se pueden incorporar sensores adicionales de alta calidad, se llega a la conclusión de que hay que utilizar técnicas adicionales a los propios sensores para garantizar unos buenos resultados en las navegaciones.

Se ha demostrado que la visión por computador es una buena técnica para conseguir los resultados deseados. Dentro de este campo, la utilización de la matriz de homografía como herramienta para realizar correcciones en los errores de orientación del robot mejora notablemente los resultados de la navegación en entornos interiores.

En entornos interiores, la extracción de contornos verticales es una buena técnica, ya que se suele disponer de un buen número de contornos que extraer, y el tiempo de cómputo es claramente inferior al de los otros extractores. Con todo, se ha observado en los experimentos que la calidad del emparejamiento no es todo lo buena que cabría esperar, y esto implica que hay que realizar un acondicionamiento del entorno para obtener buenos resultados. Como aspecto positivo de esta técnica, se ha visto que el uso de contornos verticales permite la utilización de velocidades elevadas en el robot durante las fases de aprendizaje y repetición.

Los puntos SURF permiten obtener resultados mucho más precisos para la matriz de homografía, ya que en cada imagen se extrae un mayor número de características, que además son de una calidad mayor que con los contornos verticales. Para obtener buenos resultados en las navegaciones del robot no es necesario preparar tanto el entorno; sin embargo, hay que utilizar velocidades inferiores a las empleadas con contornos verticales.

En una comparación global, se ha concluido en este proyecto que la extracción de características SURF funciona, en general, mejor que la extracción de contornos verticales, porque proporciona un mayor equilibrio entre eficiencia y robustez.

Respecto a las técnicas de emparejamiento, se han observado las diferencias obtenidas en la matriz de homografía cuando se usa, o bien una técnica de emparejamiento robusta, o bien una no robusta. En los experimentos, técnicas como RANSAC o LMS han mostrado que garantizan con una probabilidad superior al 90 % la obtención de la matriz de homografía correctamente en casos en los que el porcentaje de espurios no supere el 50 %. Para casos con un mayor número de espurios LMS desciende su eficacia drásticamente mientras que RANSAC, aunque también pierde efectividad, funciona mejor.

RANSAC es, en general, mejor método que LMS. También se ha comprobado que la solución que se obtienen con estos métodos es lo suficientemente precisa como para no necesitar realizar un posterior ajuste por mínimos cuadrados del conjunto de emparejamientos no espurios.

En la navegación, se ha observado que el robot es capaz de corregir errores de rotación no mucho mayores que 45º (unos 0.7 radianes) sin perder precisión en el recorrido.

La solución adoptada, consistente en la reparación de giros a baja velocidad y alta precisión cada vez que el robot debe cambiar de tipo de movimiento ha resultado muy beneficiosa, ya que garantiza que el robot comienza los movimientos siempre con una orientación muy precisa.

6.2. Dificultades encontradas

Durante la realización del proyecto han ido surgiendo diferentes problemas y dificultades que han tenido que ser resueltos para poder finalizarlo.

La principal dificultad ha sido la propia utilización del robot. Durante los años de estudio de la carrera, la mayoría de las prácticas y trabajos que se realizan son trabajos de diseñar aplicaciones informáticas o simulaciones en las que no interviene para nada un elemento que suministre datos del entorno. La falta de experiencia a la hora de trabajar con un sistema de estas características se ha reflejado en los primeros meses del desarrollo del proyecto.

Hay que destacar un problema que se ha tenido con el controlador de movimientos. Por defecto, las librerías de *Player* calculan las velocidades lineal y angular con una función conjunta. Esto impide al robot realizar giros con poca velocidad angular salvo que se aplique también una velocidad lineal en el desplazamiento. Esta dificultad supuso inicialmente un freno al desarrollo del proyecto, impidiendo desarrollar otras posibles mejoras

en la navegación. Gracias a los compañeros en el laboratorio de robótica se resolvió el problema encontrando la opción de configuración adecuada.

Compaginar el desarrollo del proyecto con el trabajo de cursar las asignaturas del último curso de la titulación ha supuesto una dificultad añadida, ya que durante el periodo lectivo se ha dispuesto de menos tiempo para trabajar en el proyecto. Esto ha implicado una dedicación inferior al proyecto en los primeros meses y la realización de un esfuerzo mucho mayor durante la segunda mitad del mes de junio y los meses de julio y agosto.

6.3. Trabajos futuros

Partiendo del trabajo realizado en este proyecto se plantean varias ampliaciones de gran interés.

En este proyecto el robot es únicamente capaz de reproducir un recorrido aprendido previamente. Una posible mejora sería emplear la fase de aprendizaje para generar un mapa jerárquico del entorno que permita posteriormente al robot moverse libremente dentro de dicho entorno desde cualquier parte a cualquier otra. Este problema es conocido como el problema de localización jerárquica.

Otra posible ampliación sería la utilización de la información obtenida por la visión por computador para la navegación cooperativa entre robots. La idea sería utilizar un robot, que tiene incorporada una cámara, para indicar a otros robots rutas y correcciones o bien, equipando varios robots con cámaras digitales, conseguir que naveguen en formación a partir de la información de las imágenes.

Por último se propone emplear la aplicación desarrollada en este proyecto con cámaras omnidireccionales, estudiar posibles adaptaciones de las técnicas y comparar los resultados obtenidos. Es de esperar una mejora en la navegación, ya que una cámara omnidireccional captura imágenes que abarcan 360º. La información de las imágenes omnidireccionales resulta más completa por abarcar todo el entorno del robot.

Capítulo 7
Principales hitos temporales y diagrama de Gant

En este capítulo se comentan los principales hitos temporales que han acontecido durante la realización del proyecto. En la segunda sección se muestra el diagrama de Gant, donde se puede apreciar el tiempo destinado a cada parte del proyecto y las secciones que se han ido realizando en paralelo.

7.1. Hitos del proyecto

- 13 Noviembre : Comienzo del proyecto con la lectura de artículos de investigación.
- 8 Diciembre : Primera toma de contacto con el robot.
- 25 Enero - 14 Febrero : Parada por la realización de los exámenes de Febrero.
- 4 Abril: Primera versión del código terminada.
- 17 Abril: Comienzo de los experimentos.
- 8 Junio - 22 Junio: Parada por la realización de los exámenes de Junio.
- 31 Julio: Finalización de los experimentos.
- 16 Agosto: Finalización de la memoria del proyecto.
- 24 Agosto: Depósito del PFC.

7.2. Diagrama de Gant

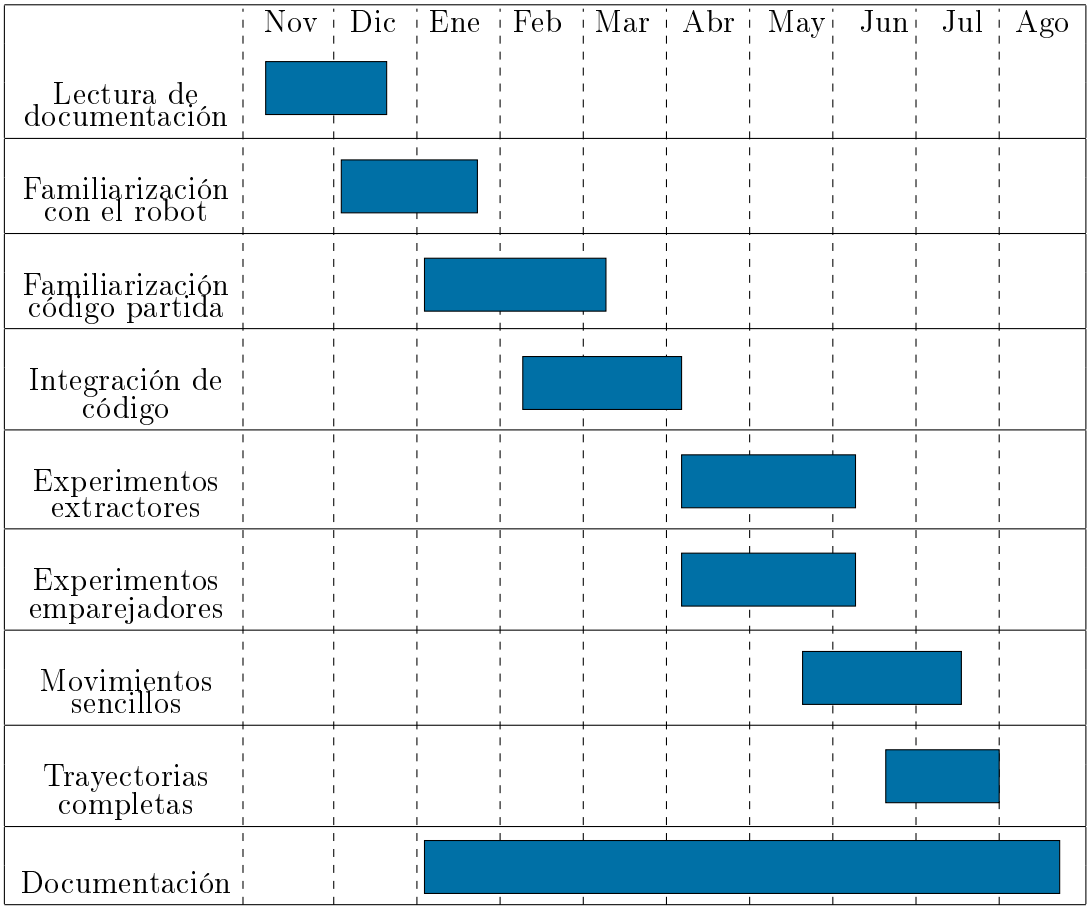

Tabla 7.1: Diagrama de Gant.

Apéndice A
Hardware de desarrollo

A.1. Hardware utilizado

Para el desarrollo del proyecto se ha empleado un robot Pioneer de los que posee el Grupo de Robótica, Percepción y Tiempo Real de la Universidad de Zaragoza. El laboratorio de robótica dispone de cuatro robots de dicha marca (figura A.1). El robot utilizado para este proyecto es, en concreto, el robot modelo *Pioneer3-dx* con las siguientes características:

- Computadora con sistema operativo *Debian*.
- 16 sensores de ultrasonido alrededor del robot
- Odómetros (encoders) asociados a las ruedas para medir la variación de movimiento.
- Sensor láser frontal para recoger información del entorno.
- Interfaz de comunicación wi-fi.
- Cámara de visión Canon VCC4

Para el proyecto no se han empleado ni los sensores de ultrasonido ni el sensor láser.

Las características hardware y las especificaciones de los modelos de robots se encuentran detalladas en la página de *Activmedia http://www.activrobots.com/ROBOTS/specs.html*.

A.2. Librerías Player

Para poder controlar el robot se han empleado unas librerías de código abierto llamadas *Player*. Estas librerías incorporan un conjunto de clases y tipos de datos que realizan el manejo del robot relativamente sencillo.

Figura A.1: Robots Pioneer del Grupo de Robótica, Percepción y Tiempo Real

Mediante un fichero de configuración indicado en la ejecución de *Player*, el programa detecta qué hardware del robot se va a emplear. En el caso de este proyecto se han utilizado el sensor de odometría y la cámara digital. Las funciones que proporciona *Player* permiten mover el robot con una determinada velocidad lineal y angular, leer datos de la odometría y capturar imágenes de forma cómoda.

Para su funcionamiento, en el robot se deben ejecutar simultáneamente dos programas. Por una parte se ejecuta el programa que controla al robot (el programado por el usuario), a la vez se tiene que estar ejecutando el programa *Player*, que se encarga de todo el control del robot.

El fichero de configuración especifica una serie de parámetros, que serán con los que trabajará el robot. Los parámetros de la cámara digital especifican el tamaño de las imágenes capturadas, si se capturan en color o en blanco y negro y el formato de la captura. Las opciones de control del movimiento contienen información acerca de la máxima velocidad permitida y el tipo de control de movimientos.

Toda la información acerca de *Player* se encuentra disponible en la página web `http://playerstage.sourceforge.net`

Apéndice B
Extractores de características en imágenes

B.1. Extracción de contornos verticales

La extracción de contornos verticales a partir de la intensidad de las imágenes es una técnica básica y poderosa para seleccionar la información contenida en una imagen. Existen dos clases de métodos para detectar los contornos: métodos basados en la agrupación de píxeles y métodos basados en modelos de brillo.

En la primera clase de métodos, los contornos generalmente se extraen detectando máximos locales del gradiente en la dirección del gradiente o detectando intersecciones con cero en la imagen de convolución empleando el Laplaciano. Una vez que los contornos se han extraído se agrupan para formar líneas.

Los contornos verticales también se pueden extraer considerando un modelo de brillo después de haber realizado una segmentación de la imagen en regiones. De los propuestos usando este modelo, el trabajo de Burns [7] es posiblemente el más importante para extraer contornos verticales. La característica más importante de este método es la organización global de la información de brillo de la imagen en regiones de soporte de recta (LSR). Se pueden citar dos ventajas de esta técnica:

- Permite obtener contornos cuando el contraste es bajo
- Permite la extracción de información acerca del brillo de la recta además de la información geométrica.

El método consta de tres etapas.

B.1.1. Segmentación

El primer paso del procedimiento es la extracción de gradientes espaciales para segmentar la imagen. Los píxeles se agrupan en regiones que presenten una dirección similar

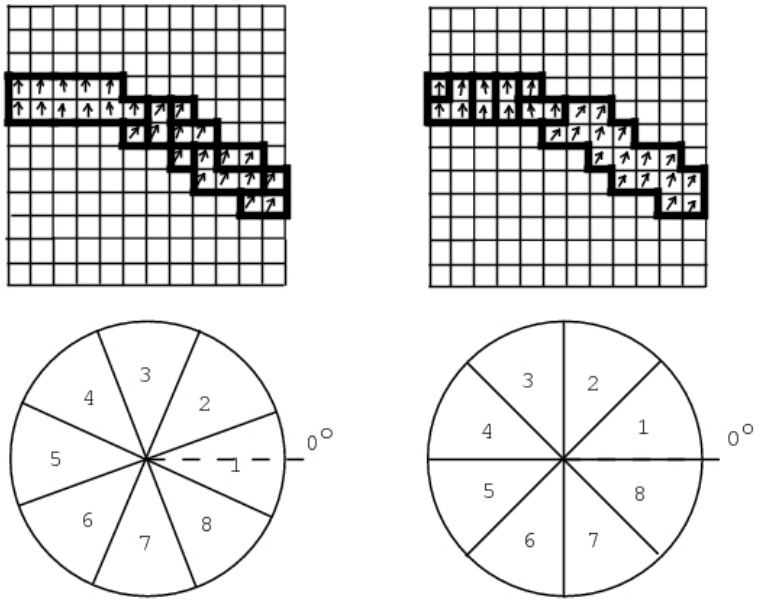

Figura B.1: Segmentación de la imagen en regiones de soporte de recta LSR.

en el gradiente del brillo, siempre que además tengan una magnitud del gradiente mayor a un umbral. Se consideran dos conjuntos solapados de particiones del espacio de direcciones; de esta forma, se evitan los problemas relacionados con la separación arbitraria de las particiones en la etapa de selección posterior (Véase figura B.1). Utilizando ambas segmentaciones se obtiene posteriormente una mejor interpretación de la agrupación de las rectas.

B.1.2. Localizando la línea

Para obtener los contornos en la imagen se ajusta una superficie de brillo plano a la LSR utilizando una aproximación por mínimos cuadrados, prediciendo el brillo (E) como una función de las coordenadas de la imagen. En este ajuste, una norma $N_w(x, y)$ proporcional a la magnitud del gradiente se utiliza para que grandes cambios en el brillo tengan mayor influencia en el ajuste. La medida minimizada a lo largo de la LSR en función de los parámetros de la superficie de brillo (A_e, B_e, C_e) se expresa como:

$$\sum_{x,y}^{LSR} [A_e\, x + B_e\, y + C_e - E(x, y)]^2\, N_w(x, y)$$

Las líneas rectas se obtienen como intersección de su plano de brillo y el plano horizontal de la media del brillo E_m en la LSR (escalada con la magnitud del gradiente).

$$E_m = \frac{\sum_{x,y}^{LSR} E(x,y)\ N_w(x,y)}{\sum_{x,y}^{LSR} N_w(x,y)}$$

La orientación de la línea se da en el rango de 0 a 2π quedando siempre el lado oscuro de la recta a la derecha. Esto es muy útil para obtener mejores emparejamientos posteriormente.

B.1.3. Atributos escogidos

A diferencia de otros emparejadores de segmentos o de puntos que solo consideran los aspectos geométricos, en este caso, el emparejamiento de los contornos rectos se realiza comparando todos los atributos proporcionados por el extractor. Este planteamiento permite emparejar características utilizando no solo la información geométrica, sino también la de la intensidad, que en muchos casos resulta muy relevante y selectiva.

El empleo de los parámetros de brillo hace que el emparejador sea mas sensible a cambios en las condiciones de iluminación, pero mejora las prestaciones del emparejador si usa solo características geométricas. Teniendo en cuenta que en este proyecto se está trabajando en entornos interiores, se puede tomar como hipótesis aceptable el hecho de que la iluminación va a presentar bastante estabilidad, lo que permite considerar los parámetros de brillo constantes en el emparejador.

La representación de un contorno recto en la imagen se compone de 6 parámetros, 4 geométricos y 2 de intensidad luminosa. Los parámetros empleados para el emparejamiento son los siguientes:

- X_m, coordenada x del punto medio del segmento
- Y_m, coordenada y del punto medio del segmento
- θ, orientación del segmento
- l, longitud del segmento
- agl, nivel medio de intensidad de la LSR
- c, contraste del contorno

B.2. SURF (Speeded Up Robust Features)

SURF es una técnica de extracción de puntos en una imagen, propuesta recientemente por Herbert Bay [6]. Se ha hecho muy famosa por la robustez de los puntos que extrae de la imagen. La principal ventaja del extractor SURF es su capacidad para extraer puntos

en una imagen sin verse afectado por rotaciones o escalados. Esto hace que el posterior emparejamiento de características entre imágenes sea muy robusto. Además lo hace con un coste computacional inferior al de otros extractores de calidad similares como SIFT [9].

A continuación se presenta una breve descripción de como funciona el extractor.

B.2.1. Puntos de interés

Lo primero que hace el algoritmo es encontrar puntos de interés dentro de la imagen. La imagen debe estar tomada en escala de grises, sin embargo, para obtener una mayor precisión numérica normaliza el valor de cada píxel, transformándolo en un valor real que se encuentre en el intervalo [0,1].

Con la matriz de píxeles ya normalizados, el algoritmo busca máximos locales del determinante de la matriz Hessiana. Dado un píxel de la imagen definido por sus coordenadas $p = (x, y)$, la matriz Hessiana de p con una escala σ se define como:

$$\mathcal{H}(p, \sigma) = \begin{pmatrix} L_{xx} & L_{xy} \\ L_{xy} & L_{yy} \end{pmatrix} \tag{B.1}$$

donde L_{xx}, L_{xy} y L_{yy} representan las derivadas segundas parciales de la gaussiana $G(\sigma)$ respecto a las coordenadas de cada píxel $p = (x, y)$.

Para que el algoritmo sea más eficiente, en lugar de calcular las segundas derivadas, se hace una aproximación mediante filtros en forma de caja, como los que aparecen en la figura B.2.

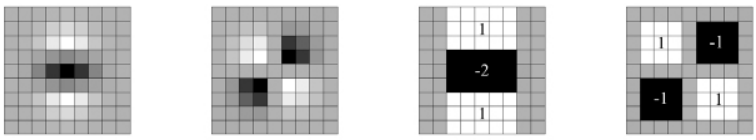

Figura B.2: De izquierda a derecha: Las derivadas parciales de segundo orden de la Gaussiana en la dirección de y en la dirección xy y las aproximaciones realizadas usando filtros de caja. Las regiones grises son igual a 0.

Si se emplean imágenes integrales, las convoluciones para estos filtros pueden ser calculadas en muy poco tiempo. Una imagen integral $\mathbf{I}_\Sigma$ es una representación de la imagen en la que cada píxel $p = (x, y)$ almacena el valor del sumatorio en x y en y desde el origen hasta el punto. El valor viene representado en la ecuación B.2

$$\mathbf{I}_\Sigma = \sum_x \sum_y p(x, y). \tag{B.2}$$

Realizando una transformación de la imagen al comienzo de la extracción para convertirla en integral, el cálculo posterior de los filtros de caja tiene coste constante independientemente del tamaño de la caja. Esto permite realizar búsquedas de máximos locales para diferentes tamaños de filtros.

B.2.2. Características de los descriptores

Para cada punto extraído en la imagen integral, se calcula un descriptor de longitud variable que identifique de forma unívoca dicho punto. Cuanto mayor sea la longitud que se emplee para los descriptores, el emparejamiento será más robusto, pero todo el proceso requerirá un mayor tiempo de CPU.

Las tres características que definen al punto son las siguientes:

- Signo del determinante de la matriz Hessiana: Se utiliza para distinguir manchas de brillo en un fondo oscuro y viceversa. Esta característica es muy interesante, ya que clasifica los puntos en brillantes y oscuros. De esta forma, en el emparejamiento, mirando el signo del determinante se podrá descartar una gran cantidad de puntos sin tener que mirar el resto de los valores del descriptor.
- Orientación del gradiente: Para cada punto detectado, el extractor define una región circular alrededor del punto y calcula la orientación dominante del gradiente dentro de dicha región.
- Sumatorio del gradiente: Se define una región cuadrada alrededor del punto seleccionado del tamaño relativo a la escala σ con la que se detectó el punto. Dependiendo del tamaño elegido para el descriptor se subdivide la región en partes iguales (4, 9, 16 ó 32) y se calculan 4 valores en cada subregión basados en los valores del gradiente obtenidos en la correspondiente subregión. El conjunto de valores obtenido constituye el descriptor del punto.

Después de realizar pruebas con los diferentes tamaños de descriptores, 16, 36, 64 ó 128, se ha elegido utilizar el de longitud 36 (aparte del signo del determinante de la matriz hessiana y la orientación del gradiente), por alcanzar un equilibrio entre la calidad que se le pide al sistema y el tiempo que necesita para realizar todos los cálculos. Además, puesto que en los experimentos que se han realizado el robot siempre se está moviendo sobre el mismo plano (el que define el suelo), el cálculo de la orientación del punto no aporta ningún beneficio, ya que la orientación de la cámara permanece constante. Eliminando el cálculo de la orientación se ha reducido el tiempo de cálculo.

Figura B.3: Ejemplo de imagen con los puntos SURF extraídos.

Además de las dos técnicas de extracción de características en imágenes utilizadas en los experimentos de este proyecto. Se han estudiado otras dos técnicas de extracción que podrían haberse incorporado al sistema. Este apéndice comenta de manera resumida en qué consisten ambas técnicas y los motivos por los que se descartó su utilización.

B.3. Detector de esquinas de Harris

Uno de los primeros métodos de extracción de puntos en imágenes es el detector de esquinas de Harris [8].

Figura B.4: Dos imágenes normales del mismo paisaje.

Este extractor se puede emplear, por ejemplo, para poder componer imágenes panorámicas mediante correspondencias entre puntos de ambas, y así luego saber por dónde

Figura B.5: Imagen panorámica compuesta a partir de dos imágenes normales.

unirlas (véanse figuras B.4 y B.5).

La idea de este extractor es mirar los cambios de intensidad que se producen en cada píxel para un tamaño determinado de ventana. En zonas lisas no se producirá ninguna variación en la intensidad del tono de gris. En el caso de un contorno, las variaciones en la intensidad seguirán todas la misma dirección. En el caso de las esquinas se producirán variaciones en la intensidad en múltiples direcciones. Con esta idea el extractor es capaz de detectar esquinas en las imágenes.

A nivel matemático, el extractor busca cambios en la intensidad de la imagen para un desplazamiento $[u, v]$ de cada píxel. El cambio de intensidad viene dado por la siguiente forma bilineal

$$E(u, v) = [u, v]\ M \begin{bmatrix} u \\ v \end{bmatrix} \tag{B.3}$$

donde M es una matriz 2x2 con la siguiente forma

$$M = \sum_{x,y} w(x, y) \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix} \tag{B.4}$$

siendo $I(x, y)$ la intensidad en el píxel de coordenadas (x, y), y w la función de ventana.

Para el estudio de las intensidades lo que se hace es analizar los valores propios λ_1 y λ_2 de la matriz M. Una esquina corresponde a aquellos píxeles para los que ambos valores propios λ_1 y λ_2 sean grandes y del mismo orden de magnitud.

El algoritmo de Harris busca puntos en los que los valores propios dan lugar a un máximo local de la llamada función de sensibilidad para esquinas, y que además superen un determinado umbral.

El problema de este método radica en que los emparejamientos entre puntos resultan muy poco robustos, en el sentido de que genera demasiados emparejamientos entre puntos que no se corresponden en la realidad.

B.4. SIFT

SIFT es una técnica precursora de SURF propuesta por David G. Lowe en 2004 [9]. Esta técnica tiene una gran aceptación por la calidad de las características que extrae.

Para detectar puntos de interés en primer lugar transforma la imagen I de la que se quieren extraer los puntos mediante matrices de convolución gaussiana a diferentes escalas σ. Posteriormente calcula diferencias D entre las distintas imágenes suavizadas.

$$L(x, y, \sigma) = G(x, y, \sigma) * I(x, y) \tag{B.5}$$

donde $*$ es la operación de convolución en x e y y

$$G(x, y, \sigma) = \frac{1}{2\pi\sigma^2} e^{\frac{-(x^2+y^2)}{2\sigma^2}} \tag{B.6}$$

A partir de B.5 se define:

$$D(x, y, \sigma) = L(x, y, k\sigma) - L(x, y, \sigma) \tag{B.7}$$

Posteriormente, el extractor calcula máximos locales en $D(x, y, \sigma)$ comparando con sus vecinos, tanto en la misma escala σ como en las escalas anterior y posterior. La figura B.6 muestra todos los píxeles con los que se realiza la comparación.

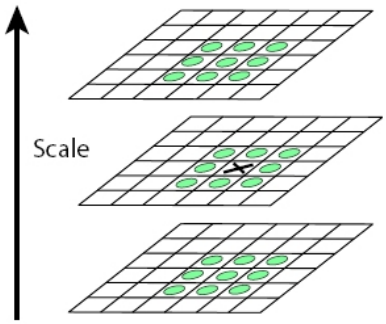

Figura B.6: Píxeles con los que se realiza la comparación del valor de D para encontrar máximos locales en SIFT.

Para todos los puntos detectados se calcula un descriptor especial empleando la magnitud y la orientación del gradiente. Partiendo de una ventana de 16x16 píxeles se calculan histogramas de orientación como los de la parte derecha de la figura B.7. Al final, para

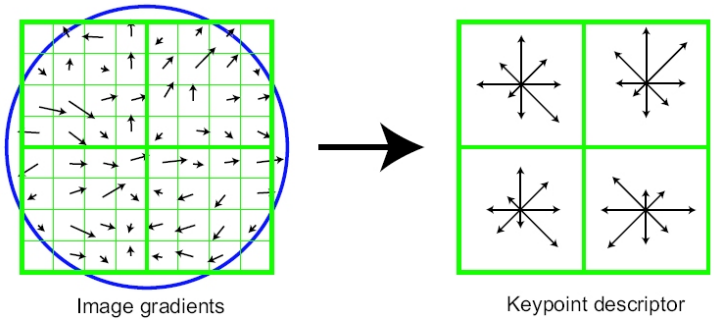

Figura B.7: Histogramas de orientación SIFT para el cálculo del descriptor. En la imagen se muestra un ejemplo de tamaño 2x2 histogramas (derecha) obtenidos a partir de una ventana de 8x8 (izquierda), el extractor utiliza una ventana de 16x16, obteniendo 4x4 histogramas de descriptores.

cada punto de interés se obtiene un descriptor de 128 parámetros de longitud (4x4x8).

Aunque la característica de este extractor es robusta e invariante respecto a cambios de escala y giros, se ha descartado debido a su elevado coste computacional. Para imágenes de 625x240, SIFT tiene un coste computacional del orden de segundos, mientras que, como se ha explicado anteriormente, el resto de técnicas tardan en extraer las características décimas de segundo en el peor caso.

Apéndice C
Técnicas de emparejamiento

Una vez que se han obtenido las características de dos imágenes, hay que buscar emparejamientos entre los conjuntos extraídos de ambas imágenes. Un emparejamiento se puede definir como una correspondencia entre una característica de la primera imagen y otra de la segunda que tiene valor del descriptor que la define similar al de la primera. Este apéndice explica las técnicas básicas que se han empleado en este proyecto para emparejar los contornos verticales y los puntos SURF extraídos de las imágenes.

Suele ocurrir que, dentro del conjunto de emparejamientos, hay un porcentaje de ellos que son espurios. Si se resuelve el sistema sobredimensionado sin eliminarlos, la solución que se obtiene no es la correcta. Para que la matriz de homografía calculada sea correcta hay que conseguir eliminar los emparejamientos espurios del conjunto de emparejamientos. En este apéndice se comentan algunas técnicas de emparejamiento de características robusto.

C.1. Emparejamiento básico de características

C.1.1. Emparejamiento de rectas

Al añadir a la información de las rectas extraídas datos acerca de su orientación exacta y del contraste que presentan dentro de la imagen, el emparejamiento de rectas resulta más eficiente. El único inconveniente de estos parámetros es que son poco robustos a cambios de iluminación de la escena.

Función de disparidad

Para poder llevar a cabo el emparejamiento, primero hay que definir una función que indique con qué grado dos rectas son semejantes. Para el emparejamiento de los contornos verticales se han empleado los argumentos presentados en [3].

La función de disparidad utilizada representa de forma inversa el grado de coincidencia de los contornos comparados. Cuanto más parecidos sean ambos contornos, menor valor devolverá esta función. La función de disparidad d viene dada por la siguiente fórmula:

$$d = \mathbf{r}^T \mathbf{S}^{-1} \mathbf{r} \tag{C.1}$$

donde $\mathbf{r}$ representa un vector con las diferencias de los parámetros de ambas rectas y $\mathbf{S}$ una matriz de covarianza que tiene en cuenta parámetros de incertidumbre del movimiento entre las imágenes y parámetros de incertidumbre relacionados con la propia extracción de las rectas.

Para determinar si dos contornos forman un emparejamiento se ha empleado el test de *chi cuadrado*. Para que dos contornos sean considerados compatibles entre sí, debe existir compatibilidad respecto a sus parámetros geométricos y también respecto a sus parámetros de brillo. En el caso de los parámetros geométricos se emplea la propia función de disparidad para realizar el test

$$d_g \leq \chi^2_4(95\,\%) \tag{C.2}$$

mientras que para los parámetros de brillo se emplea una función que tiene en cuenta las diferencias de contraste e intensidad de las rectas considerando la incertidumbre debida a cambios en la iluminación

$$d_b \leq \chi^2_2(95\,\%). \tag{C.3}$$

Para mejorar la calidad de los emparejamientos obtenidos se ha considerado la restricción de que dos rectas se emparejan si y sólo si la función de disparidad es mínima en ambas respecto al resto de las rectas consideradas. Esto quiere decir que ninguna otra recta del primer conjunto tendrá un valor menor de la función disparidad respecto a la recta del segundo conjunto y viceversa.

C.1.2. Emparejador de puntos SURF

Para realizar un emparejamiento básico de los puntos extraídos con el método SURF el proceso es secuencial. Para cada punto $\mathbf{p}_1$ de la primera imagen se buscan los puntos $\mathbf{p}_2$ de la segunda imagen que se emparejan con él. En primer lugar se eliminan como posibles emparejamientos todos aquellos puntos $\mathbf{p}_2$ que tengan signo del determinante de la matriz hessiana diferente al del punto $\mathbf{p}_1$ que se está considerando, es decir, se rechazan los puntos $\mathbf{p}_2$ que cumplen

$$\det \mathcal{H}(\mathbf{p}_1, \sigma) \det \mathcal{H}(\mathbf{p}_2, \sigma) < 0.$$

Esta primera condición elimina aproximadamente un 50 % de los puntos posibles para el emparejamiento.

Para cada uno de los puntos $\mathbf{p}_2$ restantes se calcula la norma euclídea de la diferencia de los descriptores. De todas las distancias obtenidas se almacenan las dos menores. Si la distancia más pequeña obtenida es inferior a la mitad de la segunda menor distancia, entonces se considera que hay un emparejamiento entre el punto $\mathbf{p}_1$ y el punto $\mathbf{p}_2$ para el que se ha obtenido la mínima norma de la diferencia.

Este método de emparejamiento parece peor que el utilizado para los contornos rectos ya que, a priori, nada impide que un punto de la segunda imagen pueda ser emparejado con varios de la primera. Sin embargo, se ha demostrado empíricamente que la calidad de los emparejamientos iniciales obtenidos es mucho mejor para puntos SURF con este emparejador que para contornos verticales.

C.2. Emparejamiento robusto de características previamente emparejadas

C.2.1. RANSAC

El método RANSAC (RANdom SAmple Consensus) [14] es un método de emparejamiento robusto que da como resultado la solución más votada de entre unas cuantas, calculadas a partir de conjuntos mínimos obtenidos aleatoriamente. Para calcular una homografía en dos dimensiones se necesitan un mínimo de tres emparejamientos. La idea del método RANSAC consiste en seleccionar un número de subconjuntos de tres emparejamientos elegidos aleatoriamente dentro del total, que garantice que, con una probabilidad superior al 99 %, al menos uno de dichos subconjuntos contendrá sus tres emparejamientos correctos, es decir, que ninguno de ellos es un espurio.

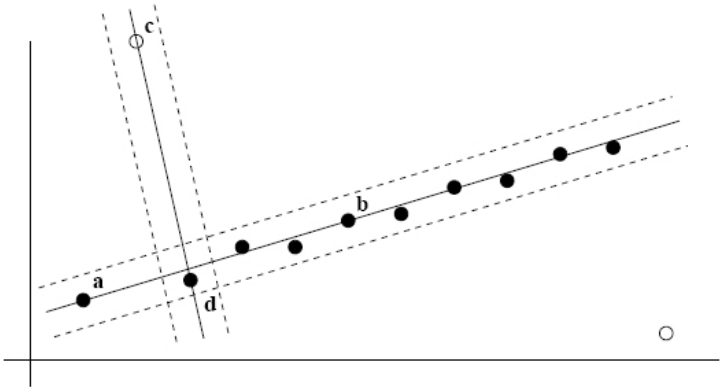

Figura C.1: Ejemplo de conjunto de puntos para ajustar a una recta.

Dados dos puntos elegidos aleatoriamente de entre todos los del conjunto, se define una recta r con ellos. Con un umbral δ elegido arbitrariamente, se consideran *inliers* todos

aquellos puntos que se encuentren dentro del intervalo $(r+\delta, r-\delta)$. La recta definida por c y d tendría únicamente 2 *inliers* mientras que la definida por a y b en la imagen tendría un total de 10 *inliers*.

Probar el total de combinaciones posibles de subconjuntos de s datos del total resultaría muy costoso computacionalmente. Si se parte de n puntos se necesitarían

$$C(n,s) = \frac{n!}{(n-s)!s!} \tag{C.4}$$

pruebas. El coste computacional de un algoritmo de estas características sería de $O(n^n)$ en el número de emparejamientos obtenidos inicialmente.

Si seleccionamos un número suficientemente grande de subconjuntos de s elementos podremos afirmar que, con una probabilidad fija, al menos habrá un subconjunto que contendrá sus s muestras dentro de la solución buena. Si encontramos la solución buena entre el subconjunto habremos hallado un algoritmo mucho más eficiente.

Número de subconjuntos a elegir

La probabilidad P con la que al menos uno de los subconjuntos seleccionados aleatoriamente esta libre de espurios viene definida por

$$P = 1 - [1 - (1-\varepsilon)^s]^m \tag{C.5}$$

donde s representa el número de elementos que debe tener el subconjunto (en el caso de las rectas 2, en el de las homografías 2D 3), ε representa una estimación del porcentaje de espurios que hay en la muestra y m el número de subconjuntos tomado.

Despejando m de la ecuación C.5 y tomando logaritmos se obtiene

$$m = \frac{log(1-P)}{log(1-(1-\varepsilon)^s)}. \tag{C.6}$$

Es interesante observar que m es independiente del número de puntos de la muestra. En la tabla C.1 se muestra el valor de m requerido para $P = 0.99$ en función de ε y s.

Para la elección de estas combinaciones se ha desarrollado una función que calcula aleatoriamente grupos de 3 elementos del conjunto de emparejamientos iniciales.

Condicionamiento de las matrices

Una vez elegida la combinación de tres emparejamientos se calcula la matriz de homografía que los relaciona utilizando las ecuaciones que se plantean en el apéndice D.

tamaño de la muestra	proporción de outliers ε						
s	5%	10%	20%	25%	30%	40%	50%
2	2	3	5	6	7	11	17
3	3	4	7	9	11	19	35
4	3	5	9	13	17	34	72
5	4	6	12	17	26	57	146
6	4	7	16	24	37	97	293
7	4	8	20	33	54	163	588
8	5	9	26	44	78	272	1177

Tabla C.1: Valores de m en función de ε y s [14].

El único problema que puede darse al resolver el sistema es que la configuración de puntos (o rectas) haya sido mala, en el sentido de que se hayan escogido tres puntos de una misma recta o tres rectas que son paralelas entre sí. Esto lleva a la obtención de una matriz con un número de condición excesivamente alto, hecho que no garantiza una solución correcta al sistema de ecuaciones. Para evitar resultados erróneos, primero se han elegido las combinaciones aleatoriamente y luego se han empleado únicamente aquellas que han producido matrices bien condicionadas. Para garantizar que se evalúan m subconjuntos, lo que se ha hecho es calcular un mayor número de combinaciones y luego se han seleccionado aquellas que han producido números de condición adecuados.

En el caso de que no hubiera suficientes matrices bien condicionadas se ha devuelto un código de error para hacer notar que no se ha podido calcular la matriz de homografía correctamente.

Elección de la mejor solución

El algoritmo calcula el número de votos o *inliers* que recibe cada solución obtenida con los subconjuntos generados aleatoriamente. Al finalizar se elige como solución aquella que tiene un mayor número de *inliers*.

Una optimización para este algoritmo es considerar como solución buena aquella que reciba más de un 90 % de votos. En caso de que una solución obtenga un gran número de *inliers* (95 %) se considera como solución final y no hace falta realizar todos los cálculos para el resto de subconjuntos.

Si se quiere obtener una solución más ajustada se puede resolver el problema sobredimensionado que generan sólo aquellos puntos que se consideran *inliers* a la homografía calculada por RANSAC.

C.2.2. Least Median of Squares (LMS)

El método de la mínima mediana de los cuadrados es otro método probabilístico de emparejamiento robusto [15] [16]. Este método da la solución de menor valor de la mediana del cuadrado de los residuos calculados para todo el conjunto de datos. Es un método muy robusto respecto a datos fuera de norma.

Igual que en el método RANSAC, inicialmente se utiliza una técnica Monte-Carlo para elegir aleatoriamente m subconjuntos de s emparejamientos diferentes. Para cada subconjunto se calcula la solución que resuelve el problema.

Este método tiene la ventaja respecto a RANSAC de no necesitar umbrales para el cálculo de buenos emparejamientos. Sin embargo, el coste computacional es mayor, ya que necesita realizar más operaciones para cada solución. Además, si el número de espurios es mayor que el 50 % la calidad de la solución desciende drásticamente. En estos casos se propone utilizar un percentil menor del residuo.

Cálculo de los residuos

En el método LMS en lugar de realizar un sistema de votaciones se calcula el valor de los residuos que genera todo el conjunto de datos respecto a cada solución obtenida. Para calcular el residuo.

Con las matrices de homografía se tiene que, en caso de ajuste perfecto:

$$\mathbf{p}_1 - \mathbf{H}_{21}\mathbf{p}_2 = 0 \tag{C.7}$$

Esto sólo se cumplirá para los emparejamientos empleados en el cálculo de la matriz de homografía. Para el resto de emparejamientos esto da

$$\mathbf{p}_1 - \mathbf{H}_{21}\mathbf{p}_2 = \epsilon \tag{C.8}$$

donde ϵ representa el vector residuo de la transformación. Tomando la norma euclídea del vector ϵ obtenemos el valor escalar del residuo que genera cada punto respecto a una matriz de homografía.

Con todos los residuos calculados para una solución se obtiene la mediana de éstos, que se emplea para calcular la bondad de la solución considerada.

Al final se elige como solución aquella que tenga un menor valor de la mediana del residuo.

Eliminación de emparejamientos espurios

A partir de la solución de mínima mediana se pueden eliminar los emparejamientos espurios. En este caso se consideran emparejamientos espurios aquellos que generen mayor residuo respecto a la solución final. En este proyecto se han considerado como buenos emparejamientos aquellos que cumplen:

$$r_i^2 \leq (2 * \hat{\sigma}^2) \tag{C.9}$$

donde r_i es el residuo que genera el i-ésimo punto respecto a la solución final y $\hat{\sigma}$ representa una estimación de la desviación estándar calculada con la siguiente fórmula:

$$\hat{\sigma} = 1.4826 * (1 + \frac{5}{n-s}) * \sqrt{M_J} \tag{C.10}$$

siendo M_J la mínima mediana, n el número de emparejamientos inicial y s los emparejamientos necesarios para obtener una solución.

Apéndice D
Cálculo de homografías

D.1. Coordenadas homogéneas

En la geometría euclídea, un punto p del plano se puede representar mediante sus dos coordenadas $p = (x, y)$. Sin embargo, cuando trabajamos con computadores aparece el problema de la discretización de los reales y el problema de la limitación en el tamaño de los números.

Para resolver el segundo problema se propuso la ampliación del número de coordenadas para representar los puntos. Añadiendo una tercera coordenada λ que represente la escala, un punto p queda definido de la siguiente manera $p = (\lambda x, \lambda y, \lambda)$.

Con esta tercera coordenada desaparece la limitación en el tamaño de las coordenadas, si λ está próximo a 0, esto querrá decir que el punto se encuentra próximo al infinito. Por definición se considera que $\lambda = 0$ implica que el punto se encuentra en el infinito. Podemos pasar de coordenadas homogéneas a coordenadas estándar a través de la forma normalizada $p = (x, y, 1)$, que se obtiene de las coordenadas homogéneas dividiendo por el parámetro λ, siempre que este no sea 0.

D.2. Matrices de transformación

Una vez que se ha explicado como se definen los puntos hay que plantear como se calculan las matrices que transforman las coordenadas.

Puesto que los puntos del plano tienen 3 coordenadas, una transformación entre dos puntos se representa mediante una matriz 3x3 de manera que:

$$\begin{pmatrix} \lambda \mathbf{x}_1 \\ \lambda \mathbf{y}_1 \\ \lambda \end{pmatrix} = \begin{pmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{pmatrix} \begin{pmatrix} \mathbf{x}_2 \\ \mathbf{y}_2 \\ 1 \end{pmatrix} \tag{D.1}$$

En las condiciones en las que se desarrolla este proyecto el movimiento va a ser siempre plano y las imágenes van a tener muy poco *baseline*, desplazamiento muy pequeño entre los centros ópticos. Partiendo de este hecho se puede realizar una simplificación en el problema eliminando de los cálculos la segunda coordenada de los puntos. En este caso la matriz de transformación pasa a tener dimensión 2x2 y el producto para transformar puntos queda como sigue:

$$\begin{pmatrix} \lambda \mathbf{x}_1 \\ \lambda \end{pmatrix} = \begin{pmatrix} h_{11} & h_{12} \\ h_{21} & 1 \end{pmatrix} \begin{pmatrix} \mathbf{x}_2 \\ 1 \end{pmatrix} \tag{D.2}$$

Esta simplificación no afecta al cálculo de errores de rotación, que son los más importantes, empleando matrices de homografía como se demuestra en [2].

El objetivo ahora es encontrar una matriz de estas características, que transforme los puntos extraídos en una imagen con los emparejamientos obtenidos de las extracciones en otra. A partir de esta matriz podremos calcular el ángulo de rotación entre las dos imágenes.

D.3. Sistema de ecuaciones

Sea $\mathbf{p}_1$ un punto extraído en la primera imagen y $\mathbf{q}_1$ su correspondiente emparejamiento en la segunda. Si denominamos $\mathbf{H}_{21}$ a la matriz 2x2 que transforma los puntos de la segunda imagen en sus emparejamientos de la primera (matriz que lleva de 2 a 1), obtenemos la siguiente ecuación:

$$\mathbf{p}_1 = \mathbf{H}_{21}\mathbf{q}_1 \tag{D.3}$$

teniendo en cuenta D.2 se extraen las siguientes ecuaciones:

$$\begin{aligned} \lambda \mathbf{x}_1 &= h_{11}\mathbf{x}_2 + h_{12} \\ \lambda &= h_{21}\mathbf{x}_2 + 1 \end{aligned} \tag{D.4}$$

Sustituyendo en la primera ecuación el valor de λ dado en la segunda se obtiene:

$$h_{11}\mathbf{x}_2 + h_{12} - h_{21}\mathbf{x}_1\mathbf{x}_2 = \mathbf{x}_1 \tag{D.5}$$

La ecuación D.5 representa la ecuación que se obtiene de un emparejamiento. Si combinamos las ecuaciones que se obtienen de tres emparejamientos diferentes se plantea el

siguiente sistema:

$$\begin{pmatrix} \mathbf{x}_{21} & 1 & -\mathbf{x}_{11}\mathbf{x}_{21} \\ \mathbf{x}_{22} & 1 & -\mathbf{x}_{12}\mathbf{x}_{22} \\ \mathbf{x}_{23} & 1 & -\mathbf{x}_{13}\mathbf{x}_{23} \end{pmatrix} \begin{pmatrix} h_{11} \\ h_{12} \\ h_{21} \end{pmatrix} = \begin{pmatrix} \mathbf{x}_{11} \\ \mathbf{x}_{12} \\ \mathbf{x}_{13} \end{pmatrix} \tag{D.6}$$

Resolviendo este sistema de tres ecuaciones con tres incógnitas obtenemos los parámetros que nos definen la matriz de homografía $\mathbf{H}_{21}$.

El parámetro que se emplea para estimar el ángulo de diferencia, como se demuestra en [2], es el valor del elemento h_{21} de la matriz. Se podrían emplear los otros elementos de la matriz; sin embargo, está demostrado que presentan una mayor sensibilidad al ruido o a errores en los cálculos, convirtiéndose por ello en peores estimadores.

D.4. Sistema sobredimensionado

Como norma general, cuando se comparan las características extraídas de dos imágenes, si estas son más o menos similares, se obtienen más emparejamientos de los tres que son necesarios para el cálculo de una homografía. En este caso el sistema de ecuaciones sobredimensionado queda como sigue:

$$\begin{pmatrix} \mathbf{x}_{21} & 1 & -\mathbf{x}_{11}\mathbf{x}_{21} \\ \mathbf{x}_{22} & 1 & -\mathbf{x}_{12}\mathbf{x}_{22} \\ \mathbf{x}_{23} & 1 & -\mathbf{x}_{13}\mathbf{x}_{23} \\ \vdots & \vdots & \vdots \\ \mathbf{x}_{2n} & 1 & -\mathbf{x}_{1n}\mathbf{x}_{2n} \end{pmatrix} \begin{pmatrix} h_{11} \\ h_{12} \\ h_{21} \end{pmatrix} = \begin{pmatrix} \mathbf{x}_{11} \\ \mathbf{x}_{12} \\ \mathbf{x}_{13} \\ \vdots \\ \mathbf{x}_{1n} \end{pmatrix}. \tag{D.7}$$

Para resolver este problema hay que recurrir a técnicas de mínimos cuadrados. En este proyecto se han empleado para obtener la solución las ecuaciones normales para un sistema sobredimensionado.

El sistema sobredimensionado tiene la forma

$$\mathbf{A}x = \mathbf{b} \tag{D.8}$$

donde $\mathbf{A}$ es una matriz nx3, x es el vector con las tres incógnitas de la matriz de homografía y b es un vector de n componentes como el de la ecuación D.7.

Para resolver este sistema mediante ecuaciones normales lo que se hace es multiplicar ambos miembros de la igualdad por $\mathbf{A}^T$.

$$\mathbf{A}^T\mathbf{A}x = \mathbf{A}^T\mathbf{b} \tag{D.9}$$

quedando de esta forma un sistema con 3 ecuaciones y 3 incógnitas como el planteado en D.6. Se puede demostrar que la solución que se obtiene para este sistema es la que cumple que la norma euclídea del residuo para el sistema sobredimensionado es mínima.

Apéndice E
Manual de usuario

Este apéndice pretende servir como manual de ayuda para nuevos usuarios de la aplicación. En él se explican todas las opciones de que dispone el programa y como utilizar cada una de ellas.

E.1. Interacción con el usuario

El software incorporado en los robots *Pioneer* no permite la programación de aplicaciones con entorno gráfico. No obstante, dada la importancia que tiene la facilidad de uso en cualquier software informático se ha pretendido que la aplicación final tuviera un entorno amigable.

Toda la interfaz de usuario gira en torno a un menú en modo texto que muestre al usuario las posibilidades que ofrece el software. La imagen E.1 muestra una vista general de la aplicación.

Todas las opciones se han numerado con códigos de uno o dos dígitos que sean claramente identificables en la pantalla de inicio.

La pantalla de inicio se ha dividido en varias secciones informativas:

- Cabecera del programa: la parte superior muestra el título del programa. Cumple una función estética para que el usuario sea consciente de que está dentro de la aplicación.
- Configuración: debajo del título se muestran las opciones que se encuentran activas en el momento actual. Se muestra el nombre de la misión, el sistema de extracción de características y el tipo de emparejamiento que se esta usando. También se indica al usuario si el programa almacenará o no las imágenes que vaya capturando.
- Opciones: muestra la lista de opciones que se pueden realizar con el programa, numeradas con códigos.

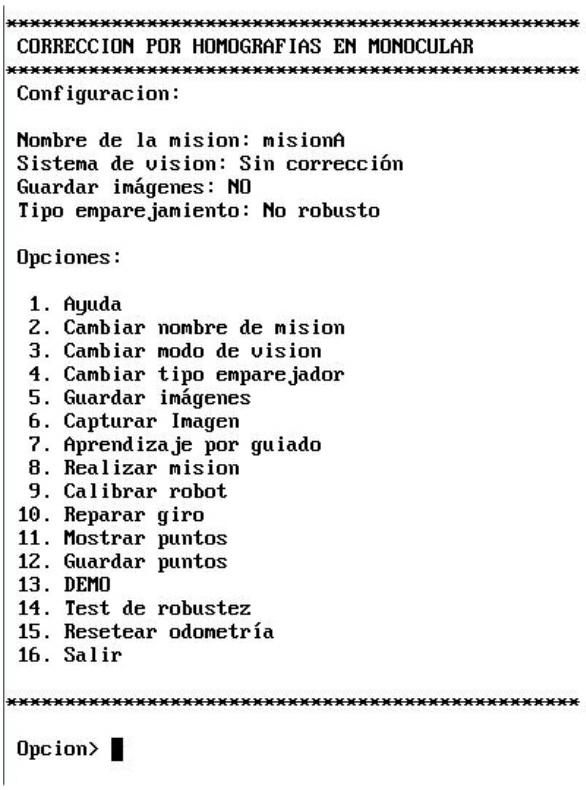

Figura E.1: Aspecto de la aplicación desarrollada.

- Consola: en esta zona de la pantalla el usuario escribe el código que desea ejecutar e introduce los datos que se le vayan pidiendo.

E.2. Ayuda interactiva

Para usuarios que empleen por primera vez la aplicación o que no sean expertos en su utilización se ha añadido una ayuda interactiva dentro del programa. De esta forma si surgen dudas durante una ejecución no es necesario acudir al manual de ayuda de este documento. Para acceder a la ayuda dentro del programa hay que teclear el código número 1.

Para no llenar la pantalla con toda la información del programa, que es muy abundante, se ha organizado la información por opciones del programa. Cuando el usuario solicita ayuda, se le pide que introduzca el código de la opción deseada y se muestran sólo los contenidos correspondientes a dicha opción. Como los códigos coinciden con los de las opciones de menú, no hay lugar a equivocaciones.

Si se requiere una mayor cantidad de información acerca del programa, en la siguiente sección se han incluido todos los detalles del funcionamiento de cada una de las opciones. De esta forma, al final, el usuario dispone de una documentación extensa y adecuada para poder utilizar el programa.

E.3. Configuración del programa

En la parte superior del menú se muestran las características que se encuentran activas en ese momento en el programa. A continuación se muestra el significado de cada una de estas líneas y como modificar la configuración del programa.

E.3.1. Nombre de la misión

El nombre de la misión indica el nombre con el que se guardarán los diferentes ficheros durante la ejecución del programa, ya sean imágenes, ficheros de características extraídas o ficheros Matlab. Si mandamos al robot que repita una trayectoria, los archivos que buscará como referencias deberán tener el mismo nombre que la misión. El usuario no deberá preocuparse por la numeración de los ficheros ya que el software la realiza automáticamente.

El nombre que aparece por defecto es *misionA*. Tecleando el código número 2 el usuario puede modificar este nombre.

E.3.2. Sistema de visión

Esta línea indica la técnica de visión que se utilizará para extraer características de las imágenes. Si no se emplea ninguna, significa que el robot sólo se moverá utilizando la odometría. Burns indica que se emplearán los contornos verticales y SURF indica que se emplearán características SURF.

Por defecto el robot navega sin utilizar la visión por computador. Mediante la opción número 3 del menú de opciones, el usuario puede cambiar las características utilizadas por el sistema de visión que empleará el robot.

E.3.3. Emparejamiento

El emparejamiento expresa cuál va a ser el emparejador que se empleará para comparar las características de las imágenes. Los tipos de extractor que pueden estar activados son los siguientes:

- Emparejamiento NO robusto
- Emparejamiento robusto por RANSAC
- Emparejamiento robusto por LMS

- Emparejamiento robusto por RANSAC + criba y ajuste por mínimos cuadrados
- Emparejamiento robusto por LMS + criba y ajuste por mínimos cuadrados

Inicialmente el robot no realiza un emparejamiento robusto para el cálculo de las homografías. La opción número 4 del menú permite modificar el tipo de emparejamiento.

E.3.4. Guardar imágenes

Está opción indica que el programa irá almacenando las fotos que vaya tomando durante sus movimientos. Las fotos que almacene tendrán dibujadas además las características extraídas. De esta forma el usuario podrá observar al finalizar la ejecución del programa las diferentes secuencias de imágenes que definen las trayectorias seguidas por el robot. Hay que tener cuidado porque el tiempo de cómputo de los algoritmos se incrementa notablemente cuando esta opción en encuentra activada, por lo que para la realización de trayectorias relativamente largas no se recomienda su activación.

Está opción puede ser interesante para valorar el posible uso de una u otra técnica dependiendo del tipo de escenario en el que se esté trabajando.

Por defecto el robot no almacena ninguna imagen. Ejecutando la quinta opción del menú (código número 5) se activa el almacenamiento de imágenes. Para volver a desactivarlo sólo hay que volver a ejecutar el código 5.

E.4. Opciones que no provocan movimiento del robot

Dentro del menú hay un conjunto de opciones que no provocan movimientos en el robot pero que pueden resultar de mucha utilidad para que la realización de misiones sea mejor. A continuación se explica el funcionamiento de estas opciones y los códigos necesarios para ejecutarlas.

E.4.1. Capturar imagen

Esta opción captura una imagen con la cámara del robot y la almacena con el nombre '*nombre_mision*'_captura_'*numero_captura*'.pgm. Si hay algún extractor de características activado durante la captura el programa, además extraerá las características elegidas y las incluirá dentro de la imagen.

Esta opción es útil antes de realizar aprendizajes ya que permite al usuario observar la orientación inicial del robot y le muestra la cantidad y calidad de las características que se van a extraer.

Si durante la fase de aprendizaje se han guardado las imágenes del recorrido, esta opción también es útil para comparar la imagen capturada al terminar una misión con la última imagen almacenada durante el aprendizaje; así el usuario puede hacerse una idea más precisa del resultado de la navegación.

Las imágenes se capturan siempre con el robot parado. Esta opción se ejecuta tecleando el código 6.

E.4.2. Calibrar robot

Aunque a esta opción se la ha llamado "Calibrar robot" en realidad no realiza una calibración. En esta opción se le pedirá al usuario que introduzca el numero de la imagen con respecto a la cual quiere calibrar el robot.La imagen tiene que haber sido aprendida durante el aprendizaje.

El programa muestra por pantalla el valor de la corrección que obtiene el robot con una imagen capturada en tiempo real y las imágenes de referencia cuyos números corresponden con el elegido por el usuario y el siguiente de la lista.

La calibración se realiza antes de poner el robot en movimiento. El código que ejecuta esta acción es el número 9.

E.4.3. Mostrar puntos

Después de realizar una misión, el programa tiene almacenado en memoria el valor de la odometría de todo el recorrido, con esta opción se muestra por pantalla toda la ruta seguida. Es una opción meramente informativa. Para ejecutar esta opción hay que teclear el código 11.

E.4.4. Guardar puntos

Con esta opción se genera un archivo de código Matlab para realizar un análisis más completo de los datos extraídos durante la misión. Ejecutando en Matlab el fichero generado, se mostrará una animación que reproduce el recorrido que ha realizado el robot según los datos odométricos. También muestra los valores de las correcciones que se han ido obteniendo.

Este fichero hace llamadas a funciones programadas en los ficheros de código de Matlab implementados en este proyecto. Por tanto, para que el código se ejecute sin errores, previamente hay que mover el archivo a la carpeta que contiene el código Matlab.

El nombre con que se almacena este fichero es '*nombre_mision*'_odometria.m. Esta opción se ejecuta tecleando el código 12.

E.4.5. Test de robustez

Para evaluar la calidad de los emparejamientos se han realizado una serie de experimentos con datos de simulación. Esta opción permite recrear dichos experimentos e incluso ampliarlos si el usuario quiere realizar mas pruebas.

Inicialmente se crea una matriz de homografía aleatoriamente. Con esta matriz se calculan conjuntos de puntos emparejados y además se introducen emparejamientos espurios. En esta opción el usuario debe introducir el tamaño de la muestra de emparejamientos que desee y el tanto por ciento de espurios que quiere que haya en dicha muestra. A continuación se calcula la homografía resultante del conjunto de datos generado. El programa mostrará los resultados de las homografías obtenidas con cada método.

Si el tanto por ciento de espurios que el usuario introduce es 100 se realizará una simulación completa, realizando el experimento descrito en el párrafo anterior para todos los porcentajes enteros de espurios posibles. Se generará un archivo de código Matlab con el nombre '*nombre_mision*'_robustez.m. Ejecutando este fichero en Matlab se mostrarán gráficas de probabilidad de resolver la homografía en función de los espurios y errores cometidos.

El código de esta acción es el número 14.

E.4.6. Resetear odometría

Entre distintas pruebas es conveniente ejecutar esta acción para que se reinicie la odometría del robot. Si la odometría no se resetea, los datos que lea el robot no tendrán ningún sentido y los resultados que se obtengan serán con toda probabilidad erróneos.

Para ejecutar esta opción hay que teclear el código número 15.

E.4.7. Salir

Opción que termina la ejecución del programa de forma segura. Esta opción se ejecuta tecleando el código 16.

E.5. Moviendo el robot

Una vez que se conocen todas las opciones previas al movimiento del robot, se procede a comentar las opciones que permiten controlar sus movimientos. Con estas opciones hay que ser especialmente precavido ya que el robot es una herramienta de trabajo muy sensible.

E.5.1. DEMO

El programa incluye una pequeña demo que muestra los distintos resultados que se obtienen con las diferentes opciones de configuración y permite realizar un movimiento sencillo del robot.

Al usuario se le pide que introduzca las velocidades lineal y angular con las que quiere que se mueva el robot. El programa capturará una imagen, realizará un movimiento con dichas velocidades y tomará otra imagen. Luego realizará extracciones y cálculos de correcciones con todas las posibilidades disponibles y los mostrará por pantalla. Es una opción pensada para usuarios que no hayan manejado nunca el robot. Además almacenará todas las imágenes capturadas con las diferentes características extraídas para que el usuario observe los tipos de características que se extraen.

Para ejecutar la demostración hay que teclear el código 13.

E.5.2. Reparar giro

Al usuario se le pide que introduzca una imagen de referencia obtenida durante un aprendizaje. El robot girará lentamente hasta situarse con la misma orientación que la imagen de referencia elegida. Está opción es útil para evaluar la corrección de giros de manera independiente al movimiento del robot.

Antes de realizar una misión es conveniente utilizar esta opción con la primera imagen de referencia, de esta forma se garantiza que el robot comenzará la misión con la misma orientación con que realizó el aprendizaje.

Para que esta opción funcione correctamente el robot debería estar situado sobre el eje de giro de la imagen de referencia. En el momento en que el robot no sea capaz de calcular la homografía, o el valor de la corrección esté suficientemente próximo a cero, dará por terminada la reparación del giro.

Esta opción se ejecuta tecleando el código 10.

E.5.3. Aprendizaje por guiado

El sistema pedirá al usuario una velocidad que se utilizará para iniciar el movimiento, y un umbral de corrección que servirá para determinar cuando se han de almacenar las características de las imágenes capturadas. En ese momento el usuario podrá mover libremente al robot por el entorno con los siguientes controles:

- TECLA w : Desplazar el robot hacia adelante
- TECLA s : Desplazar el robot hacia atrás
- TECLA a : Girar el robot a la izquierda
- TECLA d : Girar el robot a la derecha
- TECLA 1 : Capturar una imagen
- TECLA 2 : Incrementar la velocidad
- TECLA 3 : Decrementar la velocidad
- TECLA 4 : Activar/Desactivar motores
- TECLA 5 : Terminar proceso de aprendizaje

Aparte de las imágenes que quiera tomar el usuario con la tecla "1", el robot irá almacenando las características extraídas de aquellas imágenes cuyos parámetros de corrección difieran en una magnitud mayor o igual al umbral escogido. Si la opción *guardar_ imagenes* está activada, el robot almacenará también las imágenes correspondientes, si no, sólo almacenará el fichero con las características extraídas.

El aprendizaje se realiza con el código 7.

E.5.4. Realizar misión

Esta opción es la que permite al robot repetir el recorrido aprendido en la fase de aprendizaje. Para que funcione correctamente hay que cerciorarse de que el nombre de la misión es un nombre válido, es decir, deben existir los ficheros con las características de referencia correspondientes a ese nombre de misión, que el sistema de vision empleado coincide con el que se usó en la etapa de aprendizaje y que el robot está (aproximadamente) en la misma posición inicial que cuando realizó el aprendizaje.

En esta opción el usuario no intervendrá activamente ya que es el robot el que, de forma autónoma, debe ser capaz de repetir el recorrido corrigiendo los posibles errores de odometría.

Para realizar una misión, el usuario deberá teclear el código 8.

Apéndice F
Estructura del código

Para desarrollar la aplicación se ha elegido el lenguaje de programación C++. Los motivos de su elección han sido varios. El código del que se partía al comienzo del proyecto estaba desarrollado en lenguaje C y las librerías de control del robot están implementadas en C++. Además, los ficheros objeto que contienen las funciones para la extracción de características SURF también habían sido programados en C++. Eligiendo este lenguaje para el desarrollo se ha evitado tener que reescribir todo el código existente.

Para compilar el programa se ha empleado el compilador *GCC* versión 4.0.2. Como el sistema operativo sobre el que se ha ejecutado era un sistema Linux, distribución *Debian*, el programa se ha desarrollado en sistema operativo Linux, concretamente en la distribución *SUSE*, versión 10.0.

Se pretende que este apéndice sirva de referencia para futuras modificaciones en el programa. La lectura de este documento facilitará el aprendizaje de la estructura del código, la gestión de configuraciones y el material de partida.

F.1. Gestión de Configuraciones

Todos los archivos de código generado siguen el estándar del Grupo de Robótica, Percepción y Tiempo Real de la Universidad de Zaragoza. De esta forma cualquier miembro del grupo podrá comprender rápidamente el propósito de cada fichero de código fuente leyendo la información de la cabecera sin necesidad de tener que mirar todo el contenido.

El formato de las cabeceras es

```
/*****************************************************************************
 ROBOTICS GROUP
 DEPARTAMENTO DE INFORMATICA E INGENIERIA DE SISTEMAS
 UNIVERSIDAD DE ZARAGOZA, SPAIN
*****************************************************************************
```

```
 File    :  corrector.c
 Version :  v1.0
 Purpose :  Package to obtain corrections from vertical reference
            lines from images captured in real time.
 Authors :  Ruben Martinez
 Date    :  03-04-03
*************************************************************************
 Version :  v2.0
 Purpose :  Deleted obsolete functions to calculate corrections
            Added functions to get corrections from SURF points
            and vertical contours using utilidades_homografia
 Authors :  Eduardo Montijano
 Date    :  April - 2007
*************************************************************************/
```

Además de incluir el nombre del grupo de investigación, la cabecera contiene información sobre el nombre del fichero, numeración de versiones, propósito del archivo o las modificaciones, autor/es del código y fecha en que se realizó. Como en el grupo de investigación hay miembros de varias nacionalidades diferentes, los datos de la cabecera se han escrito en inglés para que todo el mundo pueda comprenderlos.

El sistema de copias de seguridad llevado a cabo ha consistido en realizar una copia a una memoria flash al finalizar el trabajo de cada día. El periodo de duración de las copias ha sido de una semana. En la carpeta copiada se almacenaban los ficheros modificados durante el día. Cada carpeta se nombraba con el nombre de la aplicación y la fecha de creación.

F.2. Estructura del programa

El código se ha estructurado en varios módulos. De esta forma ha resultado más fácil tener organizadas y separadas las diferentes funcionalidades del programa. Además esto permite la posible ampliación de las opciones del programa de forma sencilla.

El programa incluye un módulo de trabajo con matrices desarrollado por miembros del grupo. Este módulo contiene una gran cantidad de funciones para trabajar con matrices y vectores. El código de este módulo se encuentra en los ficheros *sp_ matrix*.

También incluye otro módulo para trabajar con imágenes *pgm* de forma muy sencilla. Este módulo incluye funciones de lectura y escritura de imágenes, una función de copia y funciones que permiten dibujar las características extraídas dentro de una imagen para su posterior visualización. Se han añadido al módulo inicial funciones específicas para trabajar con las imágenes obtenidas por la cámara del robot *Pioneer*; también se ha añadido un función que permite eliminar las filas pares de la imagen; de esta forma se resuelve el

problema del entrelazado.

El módulo *utilidades_homografía* incluye todas las funciones que permiten realizar un emparejamiento robusto de un conjunto de datos y el posterior cálculo de la matriz de homografía. Como los extractores de características utilizan estructuras de datos diferentes, se han incorporado al módulo funciones de conversión de los datos; de esta forma se ha conseguido separar y normalizar el cálculo de homografías de la parte de extracción de características en imágenes.

Los módulos *extractor_BURNS* y *extractor_SURF* contienen todas las funciones necesarias para realizar la extracción de contornos verticales y puntos SURF respectivamente. Cada módulo incluye además un emparejador básico de características. Al no disponer del código fuente del extractor de características SURF ha resultado imposible normalizar los tipos de datos de ambos extractores; sin embargo, empleando el módulo de *utilidades_homografía*, se ha añadido una función en cada módulo que permite el cálculo de la matriz de homografía empleando directamente los datos del extractor. Esto evita al programador tener que realizar constantemente conversiones de datos que pueden resultar liosas.

Con el objetivo de simplificar más la tarea de futuros programadores, se ha adaptado el módulo *corrector*; tanto para contornos verticales como para puntos SURF permite realizar simultáneamente emparejamiento básico, emparejamiento robusto, cálculo de homografía y obtención de la corrección en el robot. Indicando a las funciones los dos conjuntos de características a comparar y el tipo de emparejamiento robusto deseado, estas devuelven directamente el valor de corrección necesario para solapar ambas imágenes.

Los módulos *mision* y *aprender* son de más alto nivel. Estos módulos ya trabajan con el robot e incorporan implementados los algoritmos de navegación que se han explicado en el capítulo 4 de la memoria. *Aprender* contiene todo el código de la fase de aprendizaje; *mision*, además de contener el algoritmo de repetición de una ruta con secuencia de imágenes, contiene todas las funciones que permiten calibrar el robot, reparar giros, capturar una imagen con el robot y almacenar los datos de una misión en un fichero de código de matlab; también se ha incluido una función DEMO que puede servir de referencia a los programadores mostrando el funcionamiento de las llamadas a los diferentes módulos.

El código que contiene la ayuda interactiva y todo el menú de opciones se encuentra en el fichero *seguimiento.cc*, este fichero contiene también la función *main* del programa, desde donde se realizan las llamadas a todas las demás opciones del programa.

Para realizar la compilación de manera automática se ha adjuntado un *Makefile* con todas las órdenes necesarias para la compilación y el linkado de todos los ficheros y librerías que se citan más adelante.

F.3. Eficiencia de los algoritmos

Está sección pretende evaluar el coste temporal de los distintos algoritmos en función de los datos que emplean.

En cuanto a los extractores, ambos tienen un coste computacional $O(p)$, siendo p el número de píxeles de la imagen sobre la que trabajan. La diferencia en el tiempo real de ejecución la marca el número de operaciones que se realizan con cada píxel y el número de veces que hay que recorrer todos los píxeles de la imagen. SURF realiza un cálculo mucho más pesado sobre cada píxel que el que realiza el método de Burns.

Por lo que respecta a los emparejadores, el emparejamiento básico, para ambas características es del orden $O(n^2)$, con n representando el número de características extraídas en cada imagen (suponiendo que el número de características es similar en ambas imágenes).

Los algoritmos de emparejamiento robusto tienen un coste $O(m)$, siendo m el número de muestras de 3 elementos que hay que tomar sobre el conjunto de emparejamientos.

El coste de calcular la matriz de homografía es $O(k^2)$ en el número de emparejamientos. Los costes computacionales de los algoritmos de emparejamiento, emparejamiento robusto y cálculo de homografías pueden ser despreciados en comparación con el coste de la extracción de características.

Si empleamos los contornos verticales, los datos empíricos han demostrado que el robot es capaz de realizar una iteración del conjunto de operaciones (capturar imagen, extraer, emparejar, calcular homografía y corregir) aproximadamente en un cuarto de segundo. En el caso de utilizar puntos SURF esta cifra aumenta a medio segundo, aproximadamente, es decir, que es el doble de costoso.

F.4. Listado de ficheros C++

Los archivos que contienen el código C++ (ordenados por orden alfabético) de la aplicación son los siguientes:

- aprender.cc
- aprender.h
- burns.cc
- burns.h
- corrector.cc

- corrector.h
- extractor_BURNS.cc
- extractor_BURNS.h
- extractor_SURF.cc
- extractor_SURF.h
- fasthessian.h
- image.h
- imagen.cc
- imagen.h
- ipoint.h
- libSurf.a
- Makefile
- mision.cc
- mision.h
- parametros.h
- seguimiento.cc
- sp_matrix.cc
- sp_matrix.h
- surf.h
- surflib.h
- tipos.h
- utilidades_homografia.cc
- utilidades_homografia.h

F.5. Código MATLAB

Para poder visualizar de forma cómoda los resultados de los experimentos realizados, se han implementado una serie de programas en MATLAB que permiten al usuario observar gráficas de errores y resultados obtenidos, así como animaciones de las trayectorias realizadas. Todo esto se ha desarrollado a partir de los códigos de los que se disponía al principio del proyecto que permitían dibujar un robot esquemático visto desde arriba en cualquier posición y orientación dentro del plano.

Para que el usuario no tenga que preocuparse de programar en MATLAB, el programa principal incluye opciones de generación automática de código MATLAB. De esta forma, para visualizar los resultados el usuario lo único que tiene que hacer es copiar los ficheros *.m que genera el programa principal en la carpeta en que se encuentren los ficheros de MATLAB y luego ejecutarlos desde la línea de comandos del programa.

El listado de ficheros de código de MATLAB es el siguiente:

- cloc.m
- dptrans.m
- drawRobotTipo.m
- iloc.m
- iptrans.m
- NORMALIZE_PI.m
- recorrido.m

Apéndice G
Relación del proyecto con la Ingeniería Informática

En este apéndice se pretende relacionar los contenidos del proyecto con las distintas asignaturas que se han cursado durante la carrera.

El tema de este proyecto está relacionado principalmente con la visión por computador. También han sido muy importantes los conocimientos adquiridos en las asignaturas de Sistemas de Tiempo Real, Inteligencia Artificial y Control y Programación de Robots.

Todas las asignaturas de matemáticas y estadística cursadas han servido para el cálculo matricial, la comprensión de las ecuaciones relacionadas con el cálculo de las homografías y poder entender la base matemática en la que se sustentan los diferentes extractores de características.

La asignatura de Proyectos ha sido de gran utilidad durante la realización de la memoria y para toda la planificación temporal.

De manera indirecta también se han empleado los conocimientos adquiridos en Metodología de la Programación y Estructura de Datos y Algoritmos para realizar el análisis de complejidad computacional de los algoritmos utilizados y poder crear estructuras de datos adecuadas a las necesidades del problema.

Bibliografía

[1] R. Martínez-Cantín J.J. Guerrero and C. Sagüés. Visual map-less navigation based on homographies. *Robotic Systems*, 22(10):569–581, 2005.

[2] C. Sagüés and J.J. Guerrero. Visual correction for mobile robot homing. *Robotics and Autonomous Systems*, 50(1):41–49, 2005.

[3] J.J. Guerrero and C. Sagüés. Robust line matching and estimate of homographies simultaneously. In *IbPRIA, Pattern Recognition and Image Analysis, LNCS 2652*, pages 297–307, 2003.

[4] O. Pellejero, C. Sagüés, and J.J. Guerrero. Automatic computation of fundamental matrix from matched lines. In *Current Topics in Artifical Intelligence, LNCS-LNAI 3040*, pages 197–206, 2004.

[5] O. Pellejero G. López-Nicolás, J.J. Guerrero and C. Sagüés. Computing homographies from three lines or points in an image pair. *Image Analysis and Processing*, 50(1):446–453, 2005.

[6] H. Bay, T. Tuytelaars, and L. Van Gool. Surf: Speeded up robust features. In *The ninth European Conference on Computer Vision*, 2006.

[7] J.B. Burns, A.R. Hanson, and E.M. Riseman. Extracting straight lines. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 8(4):425–455, 1986.

[8] C. Harris and M.J. Stephens. A combined corner and edge detector. In *Alvey Vision Conference*, pages 147–152, 1988.

[9] D. Lowe. Distinctive image features from scale-invariant keypoints. *Int. Journal of Computer Vision*, 60(2):91–110, 2004.

[10] Z. Chen and S.T. Birchfield. Qualitative vision-based mobile robot navigation. In *IEEE Int. Conf. on Robotics and Automation (ICRA)*, pages 1702–1708, May 2006.

[11] M. Inaba Y. Matsumoto, K. Sakai and H. Inoue. View-based approach to robot navigation. In *IEEE Int. Conf. on Intelligence Robots and Systems*, pages 1702–1708, 2000.

[12] M. Inaba Y. Matsumoto and H. Inoue. Visual navigation using view-sequenced route representation. In *IEEE Int. Conf. on Robotics and Automation*, pages 83–88, 1996.

[13] J.J. Guerrero and C. Sagüés. From lines to homographies between uncalibrated images. In *Pattern Recognition and Images Analysis, IX SNRFAI*, pages 233–240, 2001.

[14] R. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*, chapter 3, Section 7, Robust Estimation. Ed. Cambridge, 2000.

[15] Peter J. Rousseeuw. Least median of square regression. In *Journal of the American Statistical Association, Vol 79, No 38*, pages 871–880, 1984.

[16] J. Rissanen. *Encyclopedia of Statistic Sciences*, chapter Minimum description Length Principle. Ed. Cambridge, 1987.